

# Descriptor Based Analysis of Digital 3D Shapes

**Welnicka, Katarzyna**

*Publication date:*
2011

*Document Version*
Publisher's PDF, also known as Version of record

# Descriptor Based Analysis of Digital 3D Shapes

Katarzyna Wełnicka

Kongens Lyngby 2011
IMM-PHD-2011-262

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

# Summary

Analysis and processing of 3D digital shapes is a significant research area with numerous medical, industrial, and entertainment applications which has gained enormously in importance as optical scanning modalities have started to make acquired 3D geometry commonplace. The area holds many challenges. One such challenge, which is addressed in this thesis, is to develop computational methods for classifying shapes which are in agreement with the human way of understanding and classifying shapes.

In this dissertation we first present a shape descriptor based on the process of diffusion on the surface of the shape – the auto diffusion function. When all heat is inserted at a single point, the function describes how much of that heat will remain at the same point after a period of time. This method allows for finding shape features at different scales related to time parameter. For instance, in conjunction with the method of Reeb graphs for skeletonization, it is an effective tool for generating scale dependent skeletons of shapes represented as 3D triangle meshes.

The second part of the thesis aims at capturing the style phenomenon. The style of an object is easily recognized by humans but a computational method for finding the style of an object is elusive. Instead of codifying the style explicitly, which can be only done within a specific context, we develop a general method for dealing with both style and function which uses the supervision provided by a set of training examples and can be evaluated using any shape descriptor, that produces dissimilarity measures between different shapes. Our methods decouple the effect of style from the effect of function and assess how suitable a descriptor is to a specific problem.

# Resumé

Analyse og processering af digitale 3D former er et betydeligt forskningsområde med mange medicinske og industrielle anvendelser samt anvendelser i underholdningsindustrien. Det er et område, der er vokset i betydning som metoder til optisk scanning har gjort opmålt 3D geometri langt mere almindeligt end tidligere. Det er også et område med store udfordringer. En væsentlig udfordring, der er et vigtigt emne i denne afhandling, omhandler udvikling af metoder der ved hjælpe af beregning klassificerer former på en måde som er samstemmende med menneskelig forståelse og klassifikation af former.

I denne afhandling præsenterer vi først en form beskrivelse, der er baseret på diffusionsprocesser på overfladen af en form - autodiffusionsfunktionen. Når varme koncentreres i et punkt beskriver autodiffusionsfunktionen hvor meget af denne varme, der er tilbage i punktet efter et et tidsinterval. Denne metode gør det muligt at finde træk ved formen afhængigt af skala som igen er givet ved tidsparameteren. For eksempel er metoden sammen med Reeb graph metoden velegenet til at finde en skeletstruktur repræsentation af 3D modeller, der er givet som trekantsnet.

Denne anden del af denne afhandling omhandler fænomenet stil. Et objekts stil bedømmes let af mennesker, men en beregningsmæssig metode til at finde et objekts stil er et vanskeligt problem. I stedet for at kode stilen eksplicit, hvilket kun kan gøres i en specifik sammenhæng, så udvikler vi en generel metode til at håndtere både stil og funktion som anvender den indlæring vi får fra et træningssæt og som kan evalueres med enhver formbeskrivelse, der giver et afstandsmål mellem forskellige former. Vores metode afkobler effekten af stil fra den som funktionen har og giver ydermere et mål for hvor egnet en formbeskrivelse er til at løse et specifikt problem.

# Preface

This thesis was prepared at Informatics Mathematical Modelling, the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree in engineering.

The thesis deals with methods of finding meaningful shape descriptors, the ones which agree with the ways humans understand shapes. A structural shape descriptor based on the process of diffusion on a shape is proposed and a method of example based style and function detection is introduced.

The thesis consists of a summary report and a collection of research papers written during the period 2009–2011, and elsewhere published (or submitted for publication).

Kongens Lyngby, September 2011

Katarzyna Wełnicka

# Papers Included in the Thesis

[5] Katarzyna Gębal, Jakob Andreas Bærentzen, Henrik Aanæs, and Rasmus Larsen. Shape Analysis Using the Auto Diffusion Function. *Comput. Graph. Forum, 28(5):1405-1413, 2009.*

[6] Katarzyna Wełnicka, Jakob Andreas Bærentzen. Tracing Reeb Graphs of the Auto Diffusion Function for Retrieval of Salient Shape Regions at their best scale. Presented as a poster on *Symposium on Geometry Processing Lausanne 2011.*

[7] Katarzyna Wełnicka, Jakob Andreas Bærentzen Henrik Aanæs, and Rasmus Larsen. Descriptor Based Classification of Shapes in Terms of Style and Function. *IMM-Technical Report-2011-17.* Contains extended version of:

[A] Katarzyna Wełnicka, Jakob Andreas Bærentzen, Henrik Aanæs, and Rasmus Larsen. Example based style classification. In Proceedings of the *Workshop on Mesh Processing in Medical Image Analysis (in conjunction with MICCAI 2011) Toronto, Canada September 18th, 2011.* Best paper award.

# Acknowledgements

I would like to express my gratitude to my main thesis supervisor Andreas whose commitment and support was extremely valuable. Andreas contributed in many ways and I am especially grateful for his enthusiasm, creative discussions, pointing out the Laplace Beltrami operator, many useful feedback comments and a lot of his time and attention. I would also like to thank my secondary supervisors Henrik and Rasmus for bringing a broader context into this research.

I am grateful to Leonidas Guibas for inspirational discussions during my foreign stay. Thanks also go to other people from the Geometric Computation Group at Stanford, especially to Andy Nguyen and Mirella Ben-Chen for collaboration on the project about consistent maps between objects.

I'd like to thank all the people from the Image Analysis and Computer Graphics group, which was a very 'hyggelige' and research stimulating environment during the time of my studies. Especially thanks to Eina Boeck, the secretary, for her kind help in many practical matters.

The 3Shape company made a room with their scanner accessible for me for around two weeks when I scanned many chess pieces. I appreciate their hospitality and patience at that time.

Thanks to colleges I have met during the last three years: Ojas Sharma, Marek Misztal, Vedrana Andersen, Vess Perfanov, Peter Stanley, Martin Ljungvist, Otto Nielsen, Antonio dos Anjos, Nader Salman and many others who contributed to many contexts of everyday life at university.

Last but not least I would like to thank my parents Ewa and Stanisław Gębal and other members of family for understanding and being there for me through many moments of my life. Special thanks to my husband Michał for his patience and support through the entire thesis period and especially its last moments, when both his love and his proofreading was a big help.

# Contents

Chapter 1

# Introduction

Shapes are everywhere around us: from particles at nanoscale, through the way our bodies look, the houses we live in, food we eat, the tableware we use for that, sofas and chairs we sit in, fonts we see while reading this text, cars we drive and all mechanical parts inside, almost everything has a shape.

Shapes also exist in our heads. From childhood we learn how to perceive shapes, how to recognize objects based on their shape, how to interact with them and even how to design and build them. We are able to think about abstractions of different shapes, and also many mathematical constructs, especially those connected to geometry, have some shapes. We can even make our subjective judgments if we like some shapes or if some shapes go together well or not.

Today, shapes can also be digitalized. Recent development of 3D scanners allowed us to be able to capture shapes directly from real objects. This enables automating many shape related tasks, which before needed to be done by hand and required long hours of tedious work. For this purpose, algorithms for dealing with shapes are being developed. Many shape descriptors which capture different properties of shapes, are proposed each year and we do not have one which is perfect and which solves all kinds of problems. It is a challenge to have methods that mimic the human way of understanding and distinguishing shapes.

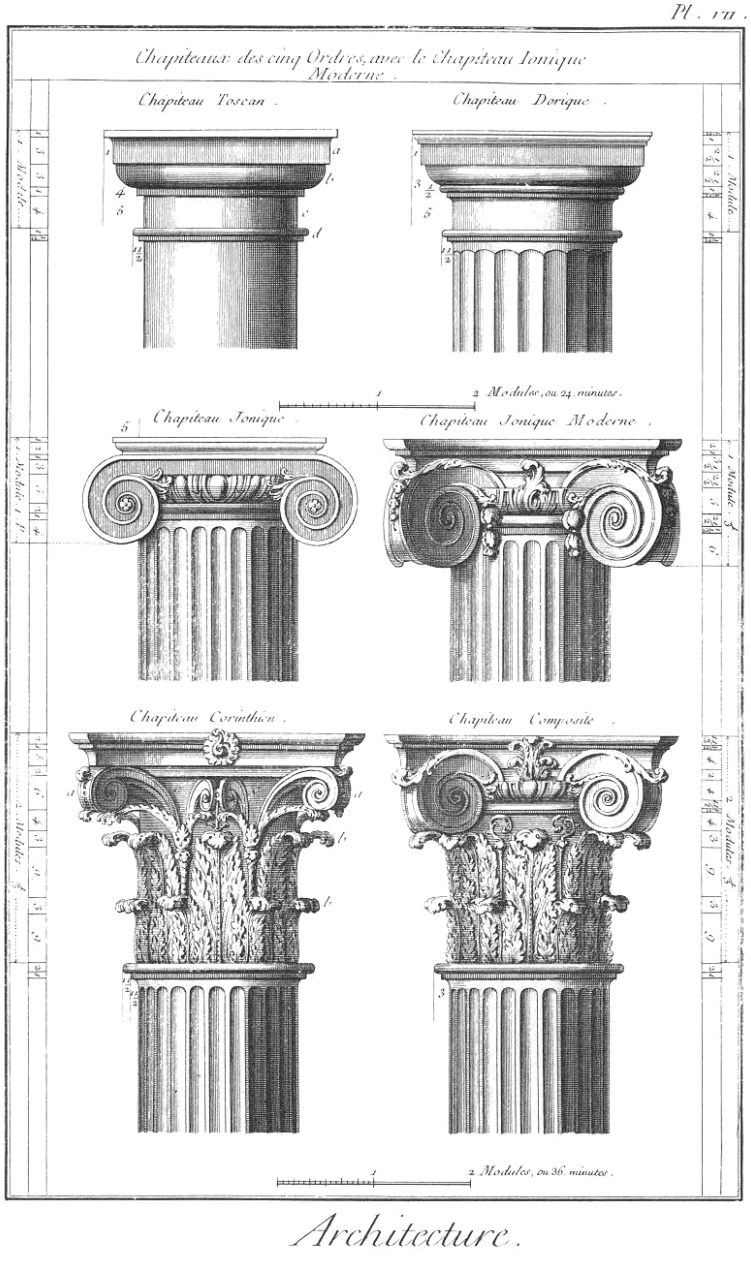



Figure 1.1: Columns from classical temple orders (source: wikipedia)

Our human way of thinking about shapes is expressed in the ability to design and to detect different styles. Styles in architecture are known from ancient times. Look for example at those classical column orders on Figure 1.1. Those styles were codified by experts and studied carefully through the centuries.

Style is obviously not only related to architecture but arises also in many other contexts. For example, we can talk about styles of tableware, furniture or fonts. Many natural phenomena also have a property that some of the shapes can be related to more than one attribute, and one of those attributes can be seen as a style, for example a human face can look different depending on the expression. Most of us can easily determine the style without being experts. For example we can say if a given set of shapes match together even if we cannot explain exactly why.

Figure 1.2: Tea set Dishwares of different styles: 'Dragonware' designed by Barbara Flügel, 'Blanche' designed by Gertrud Lönegren, transferware style from early 20th century and 'Ostfriesische Rose' by Wallendorfer Porzellan Manufaktur (source: wikipedia).

Style expresses itself as the traits of an object. However, it is hard to make a general definition of style. We understand style as a second order property of a shape while a first order property is the function. We can for example have different styles of tableware for serving tea (Figure 1.2). Each of them contains at least: a pot, a cup, a saucer, a creamer and a sugar bowl. Being a cup or a saucer is the main property and it largely determines the shape of an object.

But this property is not quite enough to define a shape, because the style also impacts how the things look. From Figure 1.2 we also see that style can be transfered from an object of one function to an object of second function, so in some way we have two orthogonal properties of shapes the stronger one, which is function and the weaker one which is the style.

Up till now most of the shape retrieval research concentrated on being able to detect the function. The frameworks which are used in order to evaluate retrieval performance of new descriptors contain different shapes classified as tables, chairs, binoculars, humans, four leg animals etc. The function is more easy to distinguish as it contains coarse geometric properties: for example the class of bikes is significantly different from the class of cars or sofas, which are in turn different from the chairs.

Style retrieval is a more complicated problem. The reason is that one needs to get rid of the function's impact on the shape. For some specific problems two orthogonal representations can be decoupled. When working with pose invariant articulated shapes, we get rid of a pose while using shape descriptors based only on the intrinsic geometry of the shapes. In a similar way by using skeletons or segmentation we can get rid of the local details, as for example the type of animal, and leave only the pose information. However, style usually influences many aspects of geometry such as proportions, local details, sharp or round edges or some additional decorations and repeating patterns.

Filtering the style, by designing a descriptors which is purely style related can be very tedious, so automatic methods are needed. Style properties also depend on the context so a set of suitable descriptors might be differ from one problem to another. In order to come up with a general style framework we have taken the example based approach, where style and function are defined by providing example shapes.

Rather than focus on the specific descriptors, we worked on methods for dealing with different descriptors in the context of style retrieval (Chapter 7). The reason for that is that it allows for a general framework, while different context is introduced by the training shapes. By working in the space of dissimilarities, we obtained independence on the nature of the descriptors which can have many different forms (for an overview of descriptors see Chapter 3).

The preliminary part of this work was to study the Laplace Beltrami based descriptors as they contain a lot of shape information. While attempting to combine the information coming from different eigenfunctions (Chapter 5) of the Laplace Beltrami operator, we arrived with a new descriptor related to the process of heat diffusion on the shape. With the use of this descriptor one is able to capture the structure of the shape at different scales (Chapter 6).

Chapter 2

# Tools

Our work is based on many methods and theories that have been invented by other people and can be found in the literature. In this chapter we present theoretical background and implementation details of main methods that were used during the project.

In Section 2.1 the Laplace Beltrami operator and heat kernel is introduced and the properties of eigenfunctions of LBO are analyzed. Section 2.2 is about Reeb graphs which allow to create a graph representation of the shape based on the level sets of some function defined in the domain of that shape. In section 2.3 we present the continuous version of translation invariant dynamic time warping method that was used for computing the distances between the outline curves and then used in the style and function related framework (Chapter 7) and section 2.4 we present slippage analysis, on which we based one the 3D shape descriptors also used as an input in the style-function setup.

## 2.1 Laplace Beltrami Operator and Diffusion

Laplace-Beltrami operator contains a lot of information about geometry of the shape. It can be found in equations describing physical phenomena such as wave propagation and heat distribution. In geometric processing it is not only used

in the context of shape analysis but also for smoothing, parametrization, editing and deformation.

The Laplace-Beltrami operator (LBO) is defined [124] on a Riemannian manifold $\Omega$ as the divergence of the gradient of a scalar function $f : \Omega \to \mathbf{R}$:

$$\Delta f = \nabla \cdot (\nabla f).$$

### 2.1.1 Eigendecomposition of LBO

Laplace Beltrami operator can be decomposed into orthogonal eigenfunctions $\phi_0, \phi_1, \ldots,$ and corresponding eigenvalues $0 \leq \lambda_0 \leq \lambda_1 \leq \ldots,$ by finding the solutions of a Helmoltz equation:

$$\Delta f + \lambda f = 0.$$

The eigenfunctions corresponding to the first few eigenvalues align surprisingly well with the protrusions and features of the shape. This phenomenon can be explained by investigating the Rayleigh Quotient method which is used for calculating the eigensolutions to the symmetric operator, which in the case of the Helmholtz equation is equal to $-\Delta$. The problem is stated as a minimization problem:

$$\phi_i = \underset{\substack{0 \neq f_i \in C_0^2(\Omega) \\ \langle \phi_{j \in \{0..i-1\}}, f_i \rangle = 0}}{\arg\min} \frac{\langle f_i, -\Delta f_i \rangle}{\langle f_i, f_i \rangle} \tag{2.1}$$

and

$$\lambda_i = \underset{\substack{0 \neq f_i \in C_0^2(\Omega) \\ \langle \phi_{j \in \{0..i-1\}}, f_i \rangle = 0}}{\min} \frac{\langle f_i, -\Delta f_i \rangle}{\langle f_i, f_i \rangle}.$$

For the compact manifold the divergence and minus gradient are formal adjoint operators [30], which means that:

$$\langle \nabla f, \mathbf{X} \rangle_v = -\langle f, \nabla \cdot \mathbf{X} \rangle. \tag{2.2}$$

Note that we use two different inner products:

$$\langle f, g \rangle = \int_\Omega f(x) g(x) dx$$

is the inner product of two scalar functions over our manifold $\Omega$ and

$$\langle \mathbf{X}, \mathbf{Y} \rangle_v = \int_\Omega \langle \mathbf{X}(x), \mathbf{Y}(x) \rangle_\Omega dx$$

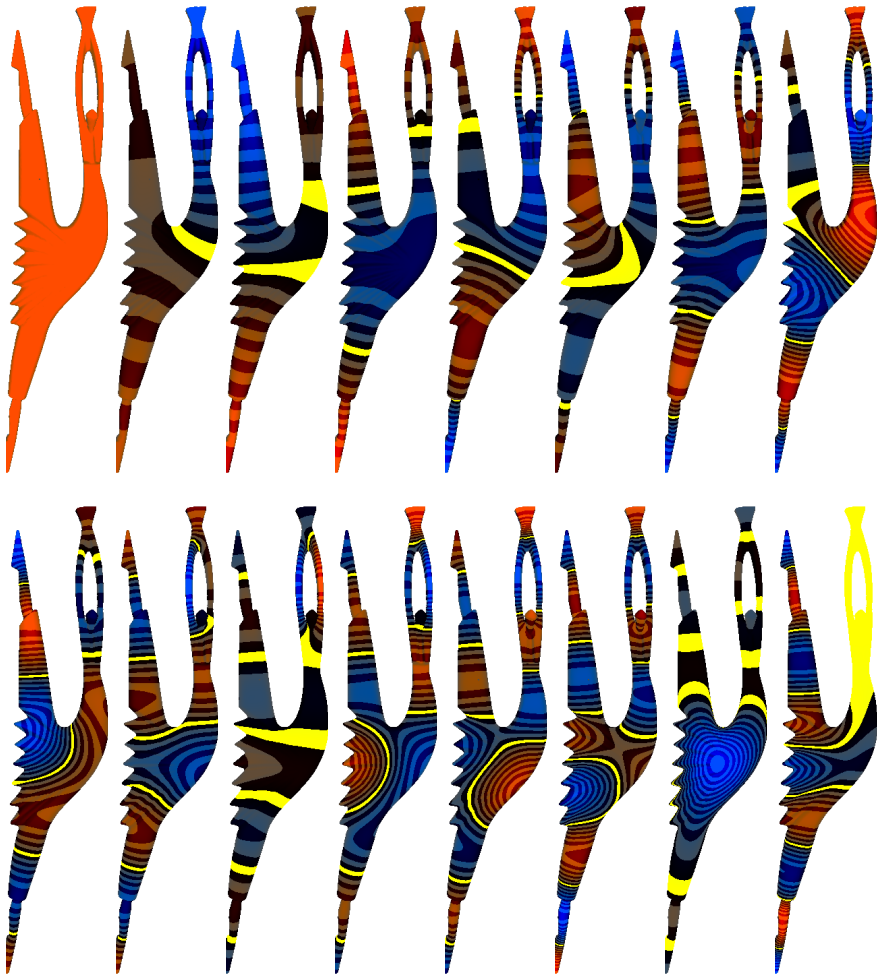


Figure 2.1: First 16 eigenvalues of the Laplace Beltrami operator for a dancer model. Red color indicates positive, blue negative, and yellow the values which are close to zero. The color pattern was striped to allow for better display of the level sets of a function. Note how functions try to 'burn' its gradient on protrusions.

is the inner product of two vector fields, where $\langle , \rangle_\Omega$ is the inner product defined by the structure of our Riemmanian manifold $\Omega$ [124].

By applying property 2.2 to $\mathbf{X} = \nabla f$ [11], we transform the 2.1 problem into

$$\phi_i = \underset{\substack{0 \neq f_i \in C_0^2(\Omega) \\ \langle \phi_{j \in \{0..i-1\}}, f_i \rangle = 0}}{\arg\min} \frac{\langle \nabla f_i, \nabla f_i \rangle_v}{\langle f_i, f_i \rangle},$$

which is equivalent to the Dirichlet energy minimalization problem with the constraint in a form of $\langle f_i, f_j \rangle = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta.

The above formulation shows an analogy between the eigenfunctions of the LBO and harmonic functions. Both minimize the Dirichlet energy, but the eigenfunctions are constrained by the orthogonality requirement rather than other boundary conditions.

Put more loosely, the eigenfunctions are mutually orthogonal, have as small as possible gradients everywhere while $\langle \phi_i, \phi_i \rangle = 1$. For a closed surface, a constant function will suffice as $\phi_0$. However, $\phi_1$, the Fiedler vector, must be orthogonal to $\phi_0$ and consequently it is positive on half the shape and negative on the other half. The requirement that the gradients should be as small as possible translates into the well known fact that the direction of change of the eigenvectors naturally follow the shape [86] as illustrated in Figure 2.1.

### 2.1.2 The Diffusion Kernel

The diffusion kernel $K(x, y, t)$, or heat kernel, is a fundamental solution to the heat equation:

$$(\Delta_x + \partial_t) u(x, t) = 0$$

where $\Delta_x$ denotes the Laplace Beltrami operator acting on the spatial variable $x$ where $t \in [0, \infty)$ is the time variable. It solves the equation with the initial condition $u(0, x) = \delta(y)$, where $\delta(y)$ is Dirac delta function at the position $y$, which means that all heat is initially concentrated in one point at the position of $y$. The general solution can be obtained by convolution of the heat kernel with the initial condition $g(x) = u(0, x)$. The heat kernel can be expressed [124, page 32] in the terms of LBO eigensolutions as:

$$K(x, y, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(x) \phi_i(y)$$

### 2.1.3 Discretization of the Laplace Beltrami Operator on a Triangular Mesh

In our implementation $\Omega$ is represented as a triangular mesh with $n$ vertices. A function defined on such a mesh has the form of an $n$ dimensional vector where each entry is related to some vertex on the mesh. A linear operator on a function has a form of $n \times n$ matrix.

Many discretization schemes of the Laplace Beltrami Operator have been proposed so far with various properties or requirements [12, 123, 155, 158].

We use the cotangent approximation of the Laplace Beltrami operator. There are at least two derivations of cotangent weights: one is based on the Discrete Exterior Calculus [39] and the other one comes from the Dirichlet energy of a map between two surfaces [118].

For a given vertex $p_i$, with a set of incident vertices $N_i$ and a function $f$ defined on vertices, laplacian has the form:

$$\nabla f(p_i) \approx \frac{1}{s_i} \sum_{j \in N_i} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} (f(p_j) - f(p_i)),$$

where angles are defined to be opposite to the edge connecting vertex $p_i$ and vertex $p_j$ (Figure 2.2) and $s_i$ is the area around the vertex, corresponding to a face being a dual of the vertex $p_i$, which means that the vertices of such face are being circumcenters of the corresponding faces. If those circumcenters are outside a triangle, the midpoint of the edge opposing $p_i$ point can be used [24].

Note that a division by $s_i$ makes this operator not symmetric. We obtain the symmetric version by applying the generalized eigenvalue formulation [125]. .

$$Lf = \lambda S f$$

Where $L$ is a matrix with entries:

$$L_{ij} = \begin{cases} -\frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2}, & \text{if } i \in N_j, \\ 0, & \text{if } j \neq i \text{ and } i \notin N_j, \\ \sum_{k \neq i} -L_{ik}, & \text{if } i = j, \end{cases}$$

and $S$ is a diagonal matrix with $S_{ii} = s_i$.

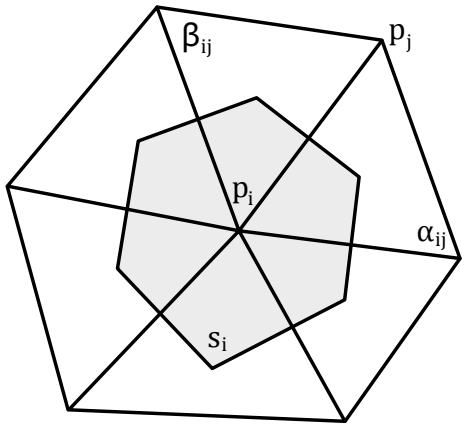



Figure 2.2: Cotan weights computation

It is worth noting that the eigenvalues solved with this system are not orthogonal with respect to the standard dot product, however they are orthogonal with respect to generalized product, where $\langle f, g \rangle = f^T S g$.

This way we obtained a sparse system. We solve a generalized eigenproblem with the $L$ and $S$ matrices. Because we are interested mostly in the first few hundreds of eigenvalues we use the sparse solver ARPACK together with SuperLU package [85].

## 2.2 Reeb Graphs

Reeb graphs [110] have its origin in differential topology and allow to extract the graph presentation from the shape. Depending on the type of the function used, different features of the shape can be outlined.

Given a manifold $\Omega$ and a function $f : \Omega \to \mathbf{R}$, $p$ is a critical point of $f$ if $\nabla(f(p)) = 0$.

$f$ is a Morse function when for each critical point $p$, the Hessian martix $H(f(p))$, which contains the second order derivatives of $f$, is non-singular. In other words we expect from the critical points of $f$ to be non-degenarate.

The Reeb graph [7] of a Morse function $f$ on a manifold $\Omega$ is the quotient space of $\Omega \times \mathbf{R}$ defined by the equivalence relation:

$$(p, f(p)) \sim (q, f(q)) \Leftrightarrow f(p) = f(q) \text{ and}$$
$$p, q \in \text{ the same connected component of } f^{-1}(f(p)).$$

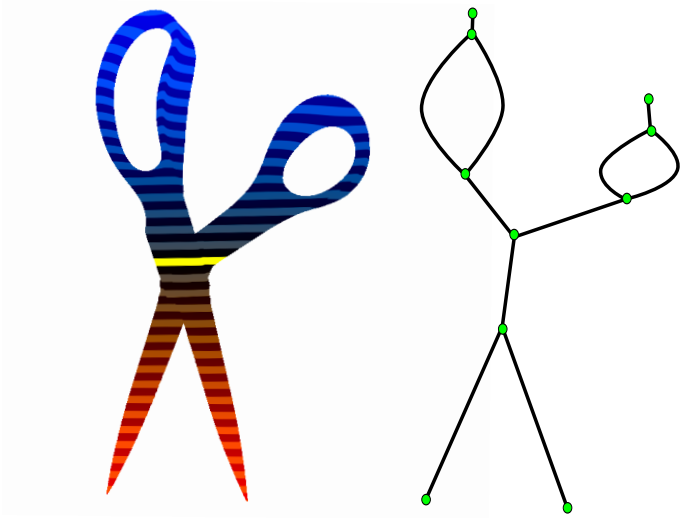


Figure 2.3: Reeb graph extracted from scissors shape when a scalar function is given as one of the coordinates.

Reeb graphs trace the evolution of the level sets of the function on the manifold. The nodes of the Reeb graph correspond to the critical points of the function $f$. At a critical point either level sets appear, disappear, split or merge. The edges are build from regular points and each of such points corresponds to one level set (Figure 2.3).

The graph itself is a topological construct, but one can position it in the space of an object by placing each of its points in the center of corresponding level set.

### 2.2.1 Reeb Graph Computation for Triangular Meshes

In order to extract the Reeb graph we use a sweep algorithm, as it allows us to compute an average position of each level set and place the skeleton points in space. The algorithm is similar to the one described by Cole-MCLaughlin et

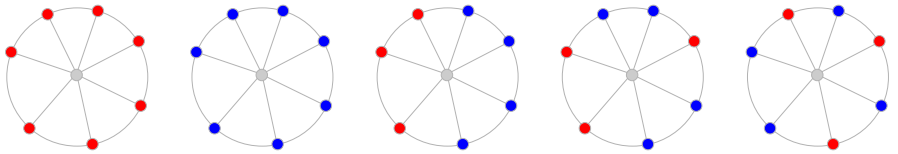



Table 2.1: The vertex with its one ring. The type of the vertex depends on the configuration of vertices with function values greater than $f(v)$, (red), and with values smaller than $f(v)$ (blue).

al. [36]. If the positioning of the vertices is not needed, a faster algorithm, that analyses the iso-contours only at the critical points [115], can be used.

In the discrete case where instead of a manifold we have a triangular mesh, the function $f$ is defined over the vertices. We assume that those values extend linearly on the edges and triangles of the mesh. In such setup $f$ is smooth and it can only have critical points at the vertices. The Morse condition can be transformed into the requirement that all critical points have pairwise different values. One way of ensuring that is to sort the vertices according to function values, and always use the order provided by that sorting. This approach might also be seen as getting rid of degeneracies by introducing $\epsilon$ perturbations to the function.

Every vertex $v$ can be classified as regular or critical being source, sink or saddle according to the function values of the vertices that belong to one-ring of that vertex. In our computations we only analyzed the cases without boundary. If boundary is needed the algorithm can be easily extended by adding more one-ring cases to be analyzed [36].

The classification depends on the number of connected segments with values greater than $f(v)$, which we denote $N_g(v)$, and the number of connected segments with values smaller than the value of $f(v)$, denoted as $N_s(v)$. Depending on these number, $v$ is:

**source** if $N_s(v) = 0$,

**sink** if $N_g(v) = 0$,

**regular** if $N_s(v) = 1$ and $N_g(v) = 1$,

**saddle** if $N_s(v) > 1$.

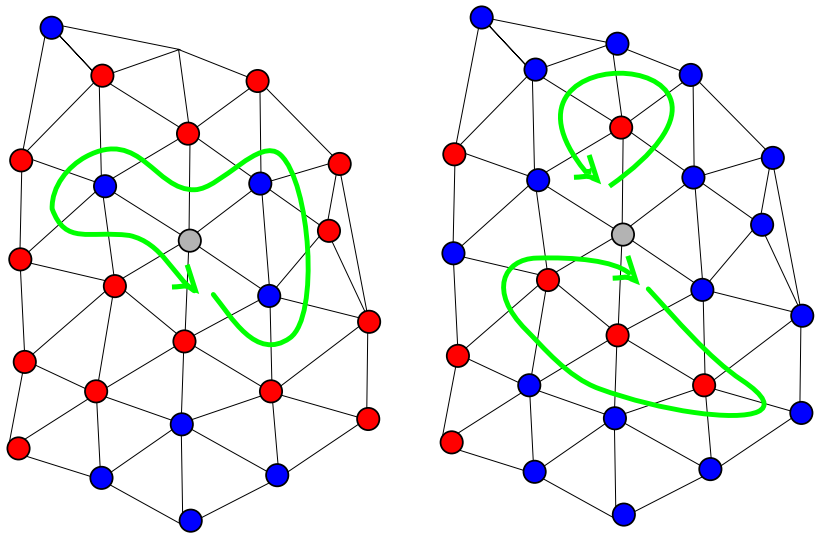


Figure 2.4: Even if one-ring of an active critical point looks the same, the global situation is different. In the first case at a saddle two loops are merged into one, second case shows that two loops are created.

From local data of one-ring we know the type of the point, however we do not know in the case of saddle what kind of topological change occur at that point, whether two level sets are being connected or whether they are being disconnected, as it requires a more a more global look. This is solved by the TraceLevelSetTriangles(t,$v_i$) method which, starting at triangle t collects the triangles while traversing the mesh. It goes from one triangle to the other through the edge having $v_k$ and $v_j$ vertices with property $k \leq i$ and $j > i$. Traverse stops when arriving back at triangle t (the process is ilustrated by green arrows on Figure 2.4). The whole sweeping procedure is presented in Algorithm 1. For the set of triangles incident to $v_i$ the following notation is used:

$T_{all}(v_i)$ contains all triangles incident to vertex $v_i$

$T_{-}(v_i)$ is the subset of $T_{all}(v_i)$ with all vertices $v_k$ such that $k \leq i$

$T_{+}(v_i)$ is the subset of $T_{all}(v_i)$ with all vertices $v_k$ such that $k \geq i$

$T_o(v_i) = T_{all}(v_i) - T_{-}(v_i) - T_{+}(v_i)$

For clarity, the Algorithm 1 returns only a list of Reeb edges. It is easy to see that from those edges the node connectivity of the whole Reeb graph can be

**Algorithm 1**: Sweeping algorithm which creates Reeb edges.

```
Data: Mesh with vertices v_i, uniquely sorted according to f value
Result: edges
activeEdges ← ∅;
edges ← ∅;
for i ← 1 to n do
    switch type(v_i) do
        case source
            e = newEdge();
            e.vstart = v_i;
            e.activeTriangles.Insert(T_all(v_i)) ; activeEdges.Insert(e);
        end
        case sink
            e = activeEdges.findEdgeWithTriangle(any t ∈ T_all(v_i));
            activeEdges.remove(e); e.vstop = v_i; edges.insert(e);
        end
        case regular
            e = activeEdges.findEdgeWithActiveTriangle(any t ∈ T_−(v_i));
            e.appendPoint(v_i);
            e.activeTriangles.Remove(T_−(v_i));
            e.activeTriangles.Add(T_+(v_i));
        end
        case saddle
            E = activeEdges.findEdgesWithActiveTriangles(T_−(v_i));
            forall e ∈ E do close old cycles
                activeEdges.remove(e); e.vstop = v_i ; edges.insert(e);
            end
            T_trans = T_0(v_i) ;
            while T_trans ≠ ∅ do create new edges
                t = T_trans.last;
                T = traceLevelSetTriangles(t,v_i);
                T_trans.remove(T);
                e = newEdge();
                e.vstart = v_i;
                e.activeTriangles.Insert(T); activeEdges.Insert(e);
            end
        end
    end
end
```

retrieved by using e.vstart and e.vstop fields of each edge e. Also it is possible to have this connectivity information being created by the time of sweeping, as it only needs a new graph node structure, and insertion operations, added when critical points are being processed.

## 2.3 Comparing Curves with Dynamic Time Warping Methods

Below we present a method of finding distances between two curves by warping one curve onto other curve.

The curve $\mathbf{X}$ is represented as polygonal chain $\mathbf{x_{i=0..n}}$. The standard dynamic time warping (DTW) problem for two curves $\mathbf{A}$ and $\mathbf{B}$ is to find a sequence of correspondences between their vertices $\mathbf{a_{i=0..n}}, \mathbf{b_{i=0..m}}$, denoted as $\mathbf{C} = \mathbf{c_{i_k j_k}}$ where $\mathbf{i_k} \in \{\mathbf{0..n}\}$ and $\mathbf{j_k} \in \{\mathbf{0..m}\}$ and satisfying conditions:

**monotonicity** if $\mathbf{c_{i_k j_k}}, \mathbf{c_{i_l j_l}} \in \mathbf{C}$ and $\mathbf{i_k} \leq \mathbf{i_l}$ then $\mathbf{j_k} \leq \mathbf{j_l}$

**continuity** for incident correspondences $\mathbf{c_{i_k j_k}}$ and $\mathbf{c_{i_{k+1} j_{k+1}}}$ we have:
$\mathbf{i_{k+1}} - \mathbf{i_k} \leq \mathbf{1}$ and $\mathbf{j_{k+1}} - \mathbf{j_k} \leq \mathbf{1}$

Classical DTW searches for the correspondence with minimal sum of lengths of vectors $\mathbf{v_{ij}} = \mathbf{a_i} - \mathbf{b_j}$ whose endpoints are defined as vertices indicated by the correspondence.

$$\mathbf{d_{DTW}}(\mathbf{A}, \mathbf{B}) = \min_{\mathbf{C(A,B)}} \sum_{\mathbf{c_{i_k j_k}} \in \mathbf{C(A,B)}} \|\mathbf{v_{i_k j_k}}\|$$

The translation invariant version of Dynamic Time Warping rather then minimizing the sum of lengths of $\mathbf{v_{ij}}$ minimizes the sum of lengths of difference of vectors $\mathbf{v_{ij}}$ for two incident correspondences:

$$\begin{aligned}\mathbf{d_{TIW}}(\mathbf{A}, \mathbf{B}) &= \min_{\mathbf{C(A,B)}} \sum_{\mathbf{c_{i_k j_k}} \in \mathbf{C(A,B)}, \mathbf{k>0}} \|\mathbf{v_{i_k j_k}} - \mathbf{v_{i_{k-1} j_{k-1}}}\| \\ &= \min_{\mathbf{C(A,B)}} \sum_{\mathbf{c_{i_k j_k}} \in \mathbf{C(A,B)}, \mathbf{k>0}} \|\mathbf{v'_{i_k j_k}}\|\end{aligned}$$

The discrete version of DTW depends heavily on how the vertices are positioned on the curve as for a given vertex the corresponding point must be taken from the vertices of the second curve.

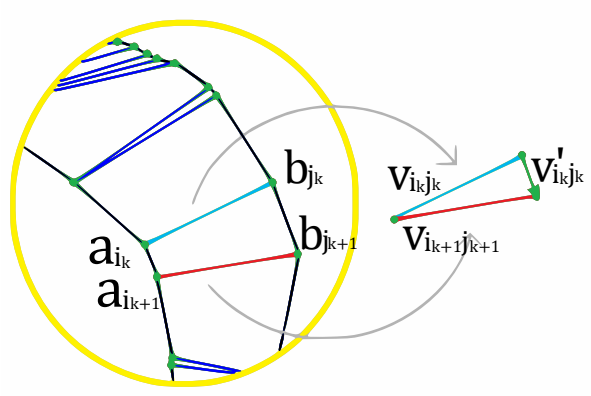



Figure 2.5: Wile the standard DTW minimizes $\mathbf{v_{i_k j_k}}$ vectors, the translation invariant version minimizes $\mathbf{v'_{i_k j_k}}$, which can be seen as it's discrete derivative.

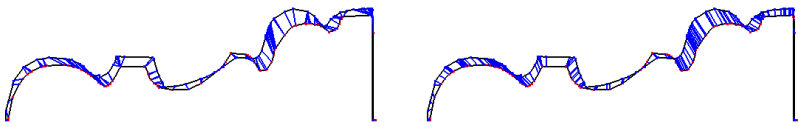


Figure 2.6: The comparison between discrete and continuous version of translation dynamic time warping. The continuous version depends only on geometry of the curve while the discrete relies on the discretization

We use the method of Efrat et al. [45] and transform the translation invariant DTW into a continuous setting. In such a case we want to represent as $\mathbf{a_i}$ any point on a polygonal chain $\mathbf{A}$ and for this purpose we extended linearly the index $\mathbf{i} \in \{\mathbf{0}...\mathbf{n}\}$ to a domain of real numbers $\mathbf{0} \leq \mathbf{i} \leq \mathbf{n}$. This is done using the interpolation of values known at vertices: $\mathbf{a_i} = (\lambda - 1)\mathbf{a}_{\lceil \mathbf{i} \rceil} + \lambda \mathbf{a}_{\lfloor \mathbf{i} \rfloor}$ where $\lambda = \frac{\mathbf{i} - \lfloor \mathbf{i} \rfloor}{\lceil \mathbf{i} \rceil - \lfloor \mathbf{i} \rfloor}$. As a result vectors $\mathbf{v_{ij}} = \mathbf{a_i} - \mathbf{b_j}$ are also extended to a two dimensional surface defined as the combinatorial manifold $\mathbf{V}(\mathbf{A}, \mathbf{B}) = \mathbf{A} \oplus -\mathbf{B}$. After this modification the translation invariant problem can be defined as finding the shortest monotonous path $\mathbf{P}(\mathbf{A}, \mathbf{B})$ on this manifold which connects the endpoints $\mathbf{v_{00}}$ and $\mathbf{v_{nm}}$. The minimized function $\mathbf{d_{CTIW}}$ is the length of this path and this value is used to establish the dissimilarity between the curves.

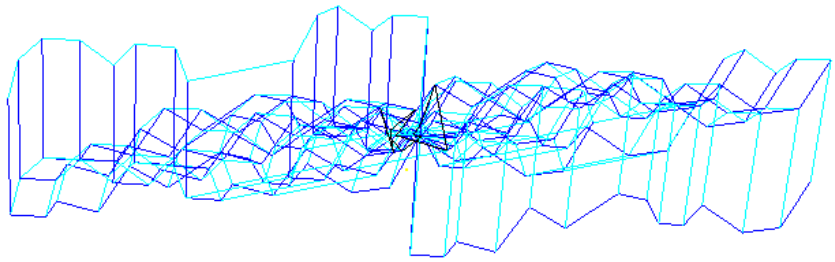


Figure 2.7: A manifold $\mathbf{V}(\mathbf{A}, \mathbf{B})$ when comparing two similar curves. Geodesic between the $\mathbf{v_{00}}$ and $\mathbf{v_{nm}}$ is marked with black. Although the embedding of this manifold is complicated, the inner structure is simple as each of the local patches are connected in a way that forms a $\mathbf{n} \times \mathbf{m}$ grid.

## 2.4 Slippage Analysis

Slippage analysis [23, 104], can be used to describe the properties of a surface patch. It is related to the point-to-plane version of the Iterative Closest Point algorithm for rigid surface alignment [34].

At one iteration step, the ICP algorithm, after finding correspondences between the points of both surfaces, searches for such transformation of the source surface, sampled at points $\mathbf{S} = \{\mathbf{s}_0, ..., \mathbf{s}_n\}$, that aligns points $\mathbf{s}_i$ with the corresponding points of the destination surface $\mathbf{d}_i$, having normals $\mathbf{n}_i$. For that reason a rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the source surface which minimizes the alignment error:

$$E(\mathbf{R}, \mathbf{t}) = \sum_i \mathbf{l}_i^2 = \sum_i ((\mathbf{R}\mathbf{s}_i + \mathbf{t} - \mathbf{d}_i) \cdot \mathbf{n}_i)^2$$

needs to be found. $\mathbf{R}$ is a $3 \times 3$ orthogonal matrix, and $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ is a vector of coordinates.

Locally, the rotation can be linearized in a way that instead of a rotation matrix $\mathbf{R}$, a vector $\mathbf{r} = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T$ of rotations around $x$, $y$ and $z$ axes is used, which leads to:

$$E(\mathbf{r}, \mathbf{t}) = \sum_i ((\mathbf{s}_i - \mathbf{d}_i + \mathbf{t}) \cdot \mathbf{n}_i + \mathbf{r} \cdot \mathbf{s}_i \times \mathbf{n}_i)^2.$$

So we search for a vector $\mathbf{x} = \begin{bmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{bmatrix}^T$ which minimizes $E$.

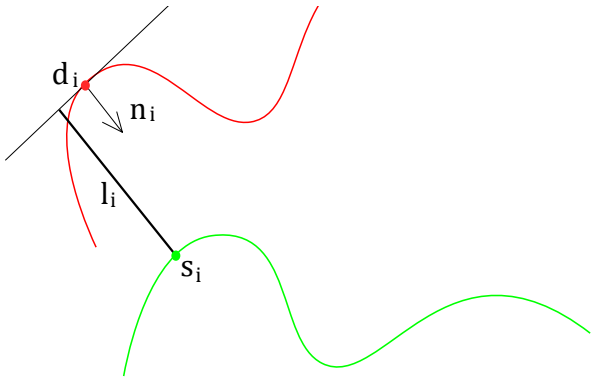



Figure 2.8: Point to plane setup. The transformation of source surface is needed which minimizes the sum of distances $l_i$.

This is achieved by setting all partial derivatives to zero $\frac{\partial E}{\partial x_i} = 0$ which results in the following system of equations:

$$\mathbf{Cx} = \mathbf{b},$$

where

$$\mathbf{C} = \sum_i \begin{bmatrix} c_{i,x}c_{i,x} & c_{i,x}c_{i,y} & c_{i,x}c_{i,z} & c_{i,x}n_{i,x} & c_{i,x}n_{i,y} & c_{i,x}n_{i,z} \\ c_{i,y}c_{i,x} & c_{i,y}c_{i,y} & c_{i,y}c_{i,z} & c_{i,y}n_{i,x} & c_{i,y}n_{i,y} & c_{i,y}n_{i,z} \\ c_{i,z}c_{i,x} & c_{i,z}c_{i,y} & c_{i,z}c_{i,z} & c_{i,z}n_{i,x} & c_{i,z}n_{i,y} & c_{i,z}n_{i,z} \\ n_{i,x}c_{i,x} & n_{i,x}c_{i,y} & n_{i,x}c_{i,z} & n_{i,x}n_{i,x} & n_{i,x}n_{i,y} & n_{i,x}n_{i,z} \\ n_{i,y}c_{i,x} & n_{i,y}c_{i,y} & n_{i,y}c_{i,z} & n_{i,y}n_{i,x} & n_{i,y}n_{i,y} & n_{i,y}n_{i,z} \\ n_{i,z}c_{i,x} & n_{i,z}c_{i,y} & n_{i,z}c_{i,z} & n_{i,z}n_{i,x} & n_{i,z}n_{i,y} & n_{i,z}n_{i,z} \end{bmatrix}$$

for $\mathbf{c} = \mathbf{s} \times \mathbf{n}$ and

$$\mathbf{b} = -\sum_i \begin{bmatrix} c_{i,x}(\mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i \\ c_{i,y}(\mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i \\ c_{i,z}(\mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i \\ n_{i,x}(\mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i \\ n_{i,y}(\mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i \\ n_{i,z}(\mathbf{s}_i - \mathbf{d}_i) \cdot \mathbf{n}_i \end{bmatrix}.$$

This equation is solved by one iteration of the ICP algorithm, and the transformation to the source surface according to values of $\mathbf{x}$ is applied.

Slippage analysis studies the situation when two copies of the same surface are optimally aligned, which means that $\mathbf{s}_i{=}\mathbf{d}_i$. In this case the matrix $\mathbf{C}$ encodes how much the error increases when moving at some direction from the solution. If the increase in all directions is large, then the problem is well defined, but if it is zero at some direction it means that this problem is not fully constrained.

In other words, we are looking at the conditioning of the alignment problem posed as the task of aligning the surface to itself, which is encoded in $\mathbf{C}$ – a Hessian matrix of that problem. This conditioning is expressed in the number of vanishing eigenvalues which is equivalent to the number of slippable motions. In practice, for the discrete setup, the eigenvalues are not likely to be perfectly zero. Instead, it is enough for eigenvalues to be smaller than some threshold. The corresponding eigenvector encodes the type of motion: translation, rotation or a mix of both. This way we can detect if the surface is:

**spherical** if there are three zero eigenvalues, and all corresponding eigenvectors are rotations,

**planar** if there are three zero eigenvalues, and one eigenvector is a rotation and the other two are translations,

**cylindrical** if there are two zero eigenvalues, with one translation and one rotation,

**revolved** if there is one rotational zero eigenvalue,

**extruded** if there is one translational zero eigenvalue,

**helical** if there is one zero eigenvalue with the eigenvector having both rotational and translational parts.

Slippage analysis can be used to describe the local properties at given point of the surface by taking samples which are within some radius from that point [104].

In order to have translational and rotational degrees of freedom measured with the same importance, vectors $\mathbf{d}_i \times \mathbf{n}_i$ and $\mathbf{n}_i$ need to have comparable magnitudes. Therefore the points need to be translated such that the center of the patch is moved into the coordinates origin, and rescaled so that the radius of the sampled patch has unit length.

Chapter 3

# Overview of Shape Descriptors

As we previously mentioned in the Introduction, style is hard to find, and not easily captured by a single descriptor. For this reason a method to combine descriptors is presented in Chapter 7. In order to be able to cast as wide a net as possible when looking for style information we need to have a broad understanding of different types of descriptors. Therefore, we present a short overview of the existing shape descriptors in the context of shape retrieval methods.

In a broad sense, a shape descriptor is information about the shape that can be automatically computed from the discrete shape representations. Descriptors vary from very simple ones being just an integer or real number, through more compound descriptors like histograms, distributions, to even more complicated as for example the graph representations or a set of images. Descriptors can also be seen as points in a space which is very often highly dimensional and nonlinear. The purpose of computing a descriptor is to classify or compare shapes in a meaningful way. Descriptors can also be used as shape fingerprints: a simplified and usually not complete representation of a shape which captures its important properties and is useful for more efficient search for similar shapes. By using local shape descriptors, we can find local correspondences between parts of shapes which can then be used in order to perform shape matching or partial shape retrieval.

In the context of shape retrieval, we can treat any descriptor as a way of comparing shapes. The simple descriptors being numbers or vectors are easy to compare. More complicated descriptors also require their own methods of assessing the similarity, as for example, methods for comparing two histograms or two distributions or graph matching. In this context, a method of comparing two shapes, even without having any intermediate representation, can also be seen as a descriptor, as it helps establish the notion of similarity between shapes.

Different descriptors capture different aspects of shapes. Some descriptors measure a specific property of the shape like, for example, orientation in space, others might be invariant to that value, and for some such a property is indirectly contained among other features of the shape. There is no single best descriptor suitable to work for all kinds of problems, rather the choice of appropriate descriptors depends a lot on the context of use. That is why this is a very active field of research and many descriptors are proposed each year. The main aim is to be able to capture the features which are also intuitive for humans and for the distances between shapes generated through such descriptors to be compatible with some aspects of similarity between the shapes.

In this survey we do not aim to present all descriptors, but to show the main methods used to obtain them. Note that some of the citations repeat in multiple sections as one descriptor can be obtained through the use of many methods.

## 3.1 Local Descriptors

Local shape descriptors are building blocks of many global methods. Usually such descriptor is a simple number or a vector evaluated at a given point. By locality we do not necessarily mean that a descriptor contains only the local information but rather it is defined on the local domain.

### 3.1.1 The Domain of Calculations

The points at which we calculate the local descriptors lie on the shape's surface or in the volume enclosed by that surface, but other options are also possible. The samples can also be taken from the surrounding space as for example in the work of Mademlis et al. [92], where the impact of the shape on the surrounding area is computed through measuring the properties of a potential field, created by the shape. Some descriptors can be evaluated at not one point, but at pairs, triplets or n-tuples of points on the shape [110].

Descriptors which require additional parameters for their computation can also be treated as local descriptors, with parameter space used instead of the shape space. Instead of shape space parameter space is being used. For example, the planar symmetry measure [77, 98] is evaluated with respect to a plane, so samples are taken from the space of all possible planes.

### 3.1.2 Simple Descriptors

The simplest local descriptors can be computed directly or can be seen as purely local because of their infinitesimal nature. The most natural example of such descriptor is a position of the point in space [110] or a normal to a surface at a given point [63]. We can also have principal curvatures or some higher degree derivatives, through the further we go the more differentiable should the surface be.

One might directly use principal curvatures $\kappa_{min}$ and $\kappa_{max}$ [156]. Another option is to use the mean $\frac{1}{2}(\kappa_{min} + \kappa_{max})$ or a Gaussian $\kappa_{min}\kappa_{max}$ curvature. Other curvature based formulas were constructed, such as the **shape index** [82] being expressed as $s = \frac{2}{\pi} \arctan \frac{\kappa_{min}+\kappa_{min}}{\kappa_{max}-\kappa_{min}}$. This measure is concentrated on the type rather than the amount of curvature at a given point.

In a similar manner, instead of coordinates and normals, other formulas which rely on this information can be applied. Simplest examples are the point's distance from the center of mass or an angle between the normal and a line connecting the center of the mass and the point's position [1].

### 3.1.3 Local Measures From Neighborhood

Very often the shape is represented as a triangular mesh or by a point set, which means that the differential values need to be computed by using discrete geometry methods. A way to establish a robust measure is to evaluate it from the neighborhood. For example, curvatures can be computed by fitting osculating polynomials to points lying nearby on the shape's surface [28]. If the shape is represented by a surface mesh, the triangle edges can be used to form the tensor of curvature, and the principal curvatures can be computed from eigenvalues of that tensor [146]. Anther approach is to take integral invariants: integrals of some functions over the intersections of surfaces (or inside of the shape) with a ball of a specific radius. Those integrals are related to many geometric invariants such as principal curvatures [120].

Neighborhood is either expressed as a clearly defined area around the point from which we are taking measurements or has a form of a kernel [93], when the impact of a given point to a measure is related to its distance from the base point. By changing the size of the neighborhood, features of different scales can be captured [120].

With the use of a neighborhood, information about the local structure of the shape can be retrieved. For example, covariance matrices of normal vectors are added with the appropriate weighing scheme to form the **normal voting tensor** [134]. The magnitudes of its eigenvalues allow to classify a vertex as a part of face, sharp edge or corner.

The **Taylor** algorithm [102] classifies each point of the surface as being part of a blend, plane, tube, cylinder, cone, a branching or a sharp vertex, by analyzing the intersection curve of the bubble centered at that point with the surface. More bubbles with different radii can be used for more exact computations or for different level of detail [101].

**Slippage analysis** [104] is related to the point to plane registration problem, and surface properties are examined through its ability to align to itself. This ability is measured by looking at the conditioning (number of nonzero eigenvalues) of the 6x6 covariance matrix of the second partial derivatives (Hessian) of the objective function, which determines the optimal rotation and translation to be applied in order to align one collection of points (with normals) to the other collection. The number of zero (or very small) eigenvalues of this Hessian indicates whether we have a surface which is: spherical, planar, cylindrical, revolved, extruded or helical.

### 3.1.4 Rich Local Descriptors

Descriptors presented in this section are also evaluated at a local point of the shape, however the output is more complex. In the broad context, local descriptors might be obtained in a similar way as global descriptors by restricting the area of computation to the neighborhood of a given point. For example Shilane et al. [133] use the spherical harmonics method [77], which was defined in the global context, but restrict the shape to regions contained in balls of different radii. Constraining to a ball centered at the sampled point is also done by Mitra et al. [99] and those regions are called **shingles**. There are also algorithms with the whole shape included as a neighborhood. In such case the advantage is to have canonical position and the canonical vector: the point of evaluation and the normal evaluated at a given point if a point is on a surface.

Johnson et al. [72] introduced **spin images**. For each local point on a surface a partial coordinate system can be defined, with position of the 3D point in the space as the center, and its normal to the surface as one of the axes. With such a frame two cylindrical coordinates are evaluated: $\alpha$: the distance to the line through the normal and $\beta$: the distance to the tangent plane. A 2D histogram called the spin image is created for each point by accumulating the $(\alpha, \beta)$ coordinates of that point in the base of the other points on the surface. Techniques from image processing can be used for further analysis. This method requires uniform sampling of the surfaces. Bin size should also be set properly, depending on the mesh resolution or sampling density.

A similar idea is used in defining the **shape context** [13] for finding the correspondence in 2D shapes. For each point a polar histogram is created by accumulating the relative polar coordinates of the other points seen in the coordinate system of the base point. $\chi^2$ statistics establishes the similarity measure between two histograms. This similarity is used as the cost in the assignment problem solved in order to get the point to point correspondences for two shapes.

Gatzke et al. [53] use **curvatures maps** as a local shape descriptor. Those are the Gaussian or mean curvatures calculated not only locally at a given point but also at an equally sampled neighborhood around that point. To get the sampling positions geodesic fans are used: first, geodesics lines are computed, equally spaced in the conformal plane of the point, and then per each geodesic the point samples on those geodesics are taken at equal distances (rings). Evaluating curvature on those samples creates a two dimensional curvature measure. To establish similarity, all possible alignments of the geodesics in fans are taken into account and the minimal dissimilarity ($L_2$ norm) is taken as the dissimilarity measure. One dimensional curvature measure can be calculated by averaging the curvature values for all samples along a ring.

Local descriptors at each point are usually used to establish the initial correspondences in a shape matching process. At a greater computational cost, rich descriptors usually contain more specified information collected from a wider neighborhood, and in result they are more powerful in reducing the ambiguity at the shape matching process.

### 3.1.5 Position Related Descriptors

There are descriptors which, although evaluated at a local position, are related to the point's position within the shape. Such descriptors are usually expensive to compute, as for any point they require the relation to all the other points of the shape expressed as a geodesic or euclidean distance or another more

complicated formula.

The idea of **local diameter** [51] is related to the diameter defined with the Medial Axis Transform [22], but it is made to be robust for meshed discrete surfaces. It is calculated based on the statistics of distances to the surface intersection with 50 rays shot from the point with a 120° opening with respect to the surface normal at the center, with the top 30% and bottom 10% outliers removed. This should to some extent be invariant to the articulated deformations, but not at places where bending occurs e.g. the elbows.

The **centricity** [62] is defined as an average of geodesic distances to all other points of the shape. The **eccentricity** [65] on the other hand is the maximum of these distances. Instead of the geodesic distance on the boundary, a geodesic distance through the enclosed volume can be used which was called the **inner-distance** by Ling et al. [89]. It is calculated using the shortest path algorithm on a graph, constructed with the sample points on the surface as vertices and with an edge connecting two points of the surface being established if it belongs to the enclosed volume.

Anther intrinsic function defined on the surface of the shape is a **conformal factor** [15]. The uniformization theorem states that any 2-manifold can be conformally mapped to a surface with the same topology having a constant Gaussian curvature. When doing such a mapping a scalar function $\phi$ describing the scaling of the metric can be found and this is exactly defined as a conformal factor. In a continuous setting $\phi$ can be computed as a solution of the non linear pde: $\nabla^2\phi = \kappa - \exp(2\phi)\kappa_u$, where $\kappa$ is the Gaussian curvature and $\kappa_u$ is the uniform target Gaussian curvature. In a discrete setup the equation $L\phi = K^u - K^o$ is solved, where $L$ is a discrete Laplace-Beltrami operator, and K are target and original Gaussian curvatures. To compare the factors for different shapes, a histogram is created and an L1 norm is used. If the shape genus is zero, the conformal factor expresses how much work is needed to transform the model into a sphere. It depends only on intrinsic properties of the shape: a local metric encoded in the Laplace-Beltrami operator and Gaussian curvature. This measure reassembles Gaussian curvature but it is much less noisy. The method is robust to the noise but sensitive to the topological changes.

Each **eigenfunctions of the Laplace-Beltrami operator** [124] can be represented as a function which minimizes Dirichlet energy [11], with the requirement that it should integrate to one, and with the additional constraint of the k-th eigenfunction being orthogonal to all i-th eigenfunctions where $i < k$. Because the gradient of a function is being minimized, the function change follows the main protrusions of the shape. Laplace-Beltrami eigenfunctions are calculated either within the domain of shape's surface manifold [86] or within the inside domain of the 3d shape [95, 122] with the Dirichlet or Neuman conditions im-

posed at the boundary. The first nonzero eigenfunction, **Fiedler vector**, is very often used as a function describing the shape [131] but also the further eigenfunctions were proposed [122]. Rustamov [125] proposes **GPS coordinates**, which for a given point are a vector of all eigenfunction values at that point divided by the square root of the corresponding eigenvalue. One should not rely on the ordering of the Laplace-Beltrami eigenfunctions because a small distortion of the shape might cause switching of them if the corresponding eigenvalues are close [69]. When shape matching is an application, 'unswitching' methods should be applied [69, 95].

The heat equation is related to the Laplace-Beltrami operator and the **heat kernel** can be expressed with the eigensolutions of the operator [124]. If all the heat was injected at a point $x$, the value of a heat kernel $K(x, y, t)$ indicates how much of that heat is present at point $y$ after time $t$ of the heat diffusion process. Note that this function is defined on two points of the manifold but by restricting $y$ to $y = x$ [54, 143] we obtain a function which expresses how much heat remains at the same point after time $t$. This function follows the general shape of the manifold. For example, heat will remain in the tip of the finger-like surface as it does not have so many options to escape, however it will spread away if the point is located on a saddle-shaped surface. The time parameter $t$ indicates the scale of features we are interested in. At small scale it is related to Gaussian curvature, according to the Minakshisundaram-Pleijel [97] expansion of the heat kernel, while at larger scale it is related to the general structure of the mesh.

## 3.2 Collecting Local Measures

When going from local to global descriptors, global information needs to be aggregated from the local points. The collecting can be done by uniform sampling. Osada et al. [110] selects points randomly based on the triangular mesh representation. First, a triangle is selected with probability proportional to its area. Then, for each chosen triangle with vertices $(A, B, C)$, a point is constructed by using two random numbers between 0 and 1: $r_1$ and $r_2$ as $P = (1 - \sqrt{r_1})A + \sqrt{r_1}(1 - r_2)B + \sqrt{r_1}r_2C$.

Very often direct integration of the discrete representation of the shape suffices. For example, if we have a triangular mesh, the measures at vertices can be taken with weights according to triangle areas incident to the given vertices [80]. Some other measures can also be taken along the edges [146] and weighted according to the edge length. For volume or embedding space related samplings, the space is divided into a three dimensional grid and values are taken from the grid

centers [95].

### 3.2.1 Integration

If a local descriptor has numerical value, integration is the simplest way to aggregate local information. Usually it means that all local values are summed up according to sampling weights. Examples of such measure may be: a center point of the shape, the shape's surface area, or volume [163] if the shape is a closed manifold. Also **moments** can be used: $m_{pqr} = \int_{boundary} x^p y^q z^r dxdydz$. By integrating the Gaussian curvature we get the Euler characteristic of the manifold according to Gauss-Bonnet theorem.

### 3.2.2 Distributions

In order to obtain more specialized information, aggregation into bins can be performed and a histogram is created. Each bin corresponds to a range of values and we accumulate how many times a given value was represented in our shape.

Binning can be seen as classifying points according to values of a local shape descriptor and then counting how much volume or area or how many sampling points each class contains. Instead of volume, other types of data can be aggregated, e.g. curvature.

Samples do not have to be single points. Instead a function can be evaluated on pairs or triples of points (see section 3.1.1). Osada et al. [110] introduces **shape distributions**. A shape signature a probability distribution sampled from some shape function. The following functions were proposed: **A3** the angle between three random points of a surface, **D1** the distance between a fixed point (for example centroid) and one random point on the surface, **D2** the distance between two random points on the surface, **D3** the square root of the area of the triangle between three random points on the surface or **D4** the cube root of the volume of the tetrahedron between four random points on the surface.

In a similar fashion, a binning space can be higher dimensional. For example Gal et al. [51] create two dimensional histograms by combining local diameter and centricity functions.

Bins can also be partitions of a sphere as in the case of the **extended Gaussian image** descriptor [63], which is a distribution of normals across the model.

In order to establish dissimilarity, a measure between different histograms needs to be taken. Osada et al. [110] gives a few possibilites: $\chi^2$ $D(f,g) = \int \frac{(f-g)^2}{f+g}$, **Minkowski** $L_n$ $D(f,g) = (\int |f-g|^n)^{1/n}$, **cumulative Minkowski** $L_n$ measures: $D(f,g) = (\int |\hat{f} - \hat{g}|^n)^{1/n}$, where $(\hat{f} = \int_{-\infty}^{x} f)$ or **Bhattacharyya**: $D(f,g) = 1 - \int \sqrt{fg}$. Ankerst et al. [5] use the extended euclidean distance $d_A^2(x,y) = (x-y) \cdot A \cdot (x-y)^T$ with weights $a_{ij} = e^{-\sigma d(i,j)}$ where $d(i,j)$ is the distance between bin centers. In order to compare two histograms, Mitra et al. use **resemblance**: the minimum and maximum histograms are computed bin-wise from two histograms and the quotient of the volume of the minimum to the volume of the maximum is defined to be the resemblance measure. Note that this value is between zero and one and has a higher value if the shapes are more similar. So to have a distance between the shapes, one minus the resemblance needs to be taken.

Quantization can cause huge histogram differences even through the shape changes smoothly because a lot of local values jump to the next bin. To combat this many of those measures take into account not only the direct difference between the bins but also the difference to a nearby bin with some weight. In order to achieve continuity between a shape's change and the descriptor, before binning the samples are convoluted with a Gaussian kernel of a fixed width [112]. This approach was also proposed within a statistics related context and named as the **density based framework** [1].

If we have a local descriptor with too many dimensions, very soon the curse of dimensionality problems to be confronted as the number of bins grows exponentially with the number of dimensions. The bag of words or **bag of features** [87] approach can be used in such case. This method was imported from other fields, appearing within text retrieval [127] and then being imported into the image [117] and video [136] retrieval problems. The first step of such approach is to construct a vocabulary by clustering (usually by using k-mean clustering) the feature space based on the samples provided from the training set. Those samples are usually taken uniformly from all of the training shapes. A bin is assigned to each cluster, which defines a partitioning of the feature space. In order to calculate the descriptor for a given shape, each bin collects the samples according to their classification. There is also a soft version of this algorithm where sample contribution to a bin is proportional to the proximity to a class center [111]. The spatially enhanced version of this algorithm also takes into account distances between two samples in the shape space, resulting in a two dimensional histogram. This can be implemented by having a second orthogonal level of clusters in the shape space, coming from the division of the embedding space [87]. Anther approach is to take pairs of samples and have one classification level according to the first sample, the other one according to the second sample and the weight on the binning being related to how far those two samples

are with respect to the shape space either by taking euclidean, geodesic, or heat kernel related distances [111].

### 3.2.3 Importance

Instead of accumulating local information, a different approach is to store a collection of local descriptors coming from the samples. Keeping all samples which were taken is memory consuming and not computationally efficient. That is why a subset of those samples needs to be selected based on feature importance. In a similar fashion, when performing binning we can incorporate importance of a sample as its weight.

We can evaluate the saliency of the sample from its local geometric properties. This means that, based on our beliefs, we heuristically design a descriptor which measures how important a sample is. Dey et al. takes $n$ most persistent maxima of the auto diffusion function at a coarse time scale [40]. In the work of Nowotni et al. [108] salient points detection is based on the 2D SIFT method which is used for finding the blob features of 2D images. Local extrema of the difference of Gaussian filters, which approximates the scale normalized Laplacian of the Gaussian, is applied to intensity image $f$: $L_{norm}(x,\sigma) = \sigma^2 \Delta G(x,\sigma) * f(x) \approx \frac{1}{k}(G(x,k\sigma) * f(x) - G(x,\sigma) * f(x))$, where $k$ denotes the fixed step scale factor. A three dimensional version of the algorithm is applied with the shape transformed into volumetric form containing ones for the cell inside and zero for the outside. A similar approach is used by Lee et al. [84] where saliency is related to the difference of a Gaussian filter with sigma as parameter of the mean curvature and the same filter at two times sigma scale. In order to accumulate information from different scales, a non-linear suppression operator can be used [66] that adds together the normalized version of $L_{norm}$s for different scales, multiplied by a square of the difference of maximum and average value.

Instead of feature points, regions can be obtained by growing techniques while fitting a quadric [50]. This allows for region related measures which are combined into the saliency formula: the area of the region, the number of local minima and maxima of the Gaussian curvature, the integral of the whole curvature and curvature variance in the region.

Saliency information can also be built relying on the performance input, supplied by a training dataset. Such a set, with additional labels indicating membership in a specific class, is used by Shilane et al. [132] to define saliency as the retrieval performance within the database of the local shape descriptor. For the query shape $X$ one evaluates the distance from all local shape descriptors $x \in X$ to all the other local descriptors from the shape $Y$ and the minimum running over all

the samples of $Y$ is taken as the distance form $x$ to $Y$. This distance is used to create a retrieval list for the descriptor at point $x$. Then, retrieval performance is measured according to Discounted Cumulative Gain (DCG). To estimate the performance measure for descriptors of a query shape, whose class label is unknown so retrieval performance cannot be computed directly, a mapping from likelihood to DCG is performed. Likelihood is defined as a multivariate normal distribution of the samples in the feature space following the idea that features which are different from the standard ones are more significant [73]. The mapping function is learned in the training phase with likelihood function evaluated based on training samples. As experimental results have shown, clustering according to likelihood groups the features with similar DCG. So for a new feature the likelihood is evaluated, then features from the training dataset having similar likelihoods can be taken and their DCG values used to asses DCG of the new feature. Funkhouser et al. [49] applies the retrieval performance idea to shape matching, but the nearby salient features are eliminated by using a distance threshold. The features are added into a list according to their performance measures, but those which are less distant to features already in the list are not included.

The connection of significance and the distribution of features was directly used by Gelfand et al. [55], where samples are binned based on a real valued local descriptor and the samples from least populated bins are taken as the most significant ones.

### 3.2.4 Space Division

A specialized description can be achieved if the integration is performed per some part of the shape, created either by dividing the embedding space or by some other clustering methods. For example, horizontal division of the space results in a descriptor informing how much of the curvature is allocated at the lower part of the shape vs. the upper part.

For comparing molecules Ankerst et al. [5] use histograms where the bins are defined as specific sections of the 3D shape: **shells** as divisions along distance from the center, **sectors** as division related to spherical angles, and **spiderweb** as both types of division combined. The amount of volume of the shape within the sector is used as the integrated value.

Bustos et al. [26] divide the shape according to octant based partitioning and add to a global shape descriptor eight sub shape descriptos computed by constraining the main point.

## 3.3 Pose Normalization

An **invariance property** term is always paired with some transformation being performed on the shape. A descriptor having invariance property will not change after applying such transformation. Different transformations are considered in this context such as: scaling, rotation, translation, or different isometric embedding in a space. It can also be a local topology change as for example connecting nearby legs of the Homer shape by a small tube [125]. Coarse scale shape descriptors might be invariant to a transformation which changes local details, such as smoothing or adding noise.

Many descriptors have invariance properties inherited form the way they were constructed. A histogram of D2 distances between sampled pairs of points is translation and rotation invariant, and one which is computed only on geodesic distances on the surface will be pose invariant.

Changing descriptors from a not-invariant to an invariant version can be done by transforming the shape into a canonical pose before applying the descriptor extraction. For example, translational invariance is achieved by calculating the center of mass and translating the shape so that the center is being in the origin of the coordinate frame. The rotation related canonical pose can be obtained by applying a spherical axis transform [5], so the eigenvectors of a covariance matrix are used as a canonical coordinate frame. This approach is also known as principal component analysis based rotation normalization [26, 154]. Kazdhan et al. [78] factors out anisotropy by rescaling the model by the inverse square root of the covariance matrix. Note that all covariance related methods require translation normalization first.

Multidimensional scaling based on geodesic distances between points located at the shape is a broadly used method of achieving a pose invariance [46, 75]. Also spectral embedding in the coordinates of first eigenfunctions of the Laplace Beltrami operator can be applied [68].

There are some issues concerning how the canonical positions are defined. For example a center of mass might not really be what is supposed to be the center of a shape, as it is sensitive to outliers – for example, a spherical shape which needs to be compared with a very similar spherical shape with a handle. Similar issue happens when applying canonical rotation based on the PCA eigensolutions. For some classes of shapes, a symmetry might be a better hint on how to position the shape [119]. For box-like shapes a rectilinear measure [88], which is related to the area of the surface and the sum of three orthogonal projected areas of the surface. The coordinate system for which the shape reaches top rectilinear scores can be chosen as the frame. Scale invariance is achieved by scaling the

shape according to its size. But it is not clear if the size is related more to the radius of the bounding sphere, the maximum distance between all pairs of points or to the variance of equally sampled surface points.

Another approach is to generate many instances of a shape, each one in a slightly different size or rotation angle, and evaluate the distance as the minimum distance [32, 53] between all possible configurations. Such an approach is also reflected in a formulation of the Gromov-Hausdorf distance [100] when the infimum is taken through all possible isometric embeddings of compared shapes. Depending on the richness of configuration space, the 'all-possible' formulation can be too expensive or even impossible to implement.

## 3.4 Harmonics

If a descriptor is a function $f$ defined in the domain $\Omega$, it can be transformed into a different representation by choosing the basis of orthogonal functions $\psi_i$ for $\Omega$, and projecting $f$ on those bases $\mu_i =< f, \psi_i >$. The most known bases for this operation for $\Omega$ being a riemanian manifold are the eigenfunctions of Laplace Beltrami operator on $\Omega$. Those functions have an advantage that if we want to go back to the original representation and take a sum $\sum_i \mu_i \psi_i$ with a finite number of $i = 0..n$, the result is a smooth approximation of $f$ and the exactness of this approximation is related to $n$ [124]. For some domains $\Omega$ the eigenfunctions can be computed analytically. If we have a circle $\Omega = S^1$ we obtain the Fourier transform, with a sphere we will get spherical harmonics. For a shape with an unknown analytic formula, an approximate solution can be found numerically [123].

### 3.4.1 Spherical Harmonics and Rotation Invariance

Spherical harmonics $Y_l^m$ are the eigenfunctions of Laplace Beltrami operator on the sphere. They are used extensively because of the possibility to group the expansion coefficients in a way which results in rotationally invariant descriptors. Each frequency $l$, or in other words eigenvalue, has a $2l + 1$ dimensional space of corresponding eigenfunctions. When we take the expansion of contribution to f according to frequency $f_l = \sum_{m=-l}^{m=l} \mu_{lm} Y_l^m$ and apply the $L_2$ norm to $f_l$ we will get the energy of a spherical component at a frequency $l$ and is rotationally invariant. A collection of such norms ordered by $l$ is called a **spherical harmonics shape descriptor** [48].

This descriptor can be used directly with any descriptors in a form of spherical functions. Funkhouser et al. [48] subdivide the space into a set of concentric shells and a spherical function is obtained by intersecting a shell of a specific radius with the voxelized boundary of a shape: in the voxel space we have 1 if a voxel contains a boundary and 0 otherwise. The side effect of the use of concentric shell division is that they are unable to detect if the inner part of the shape was rotated [77].

Vranic et al. [154] also use concentric spheres with different radii, but the function value is defined as the distance between the center of a shape and the intersection point of the shape with a ray shot from the center to given direction. If an intersection within the given shell interval does not occur, the value is set to zero.

Kazdhan et al. [79], among different possibilities, proposes the use of a spherical harmonic descriptor on extended gaussian image [63], the spherical extent function, which associates with the ray through the origin the most distant intersection with the shape, the amount of surface area that sits over the ray or the average distance and standard deviation of points on the intersection of the surface and the ray.

**Zernike moments** are an extension of spherical harmonics in a way that the radial values are also taken into account. Novotni et al. [109] use the Zernike descriptors directly on the volumetric representation of the boundary of the shape.

### 3.4.2 View Based Descriptors and Fourier Transforms

A three dimensional version of fast Fourier transform can be applied directly to a voxelized axis aligned representation of the shapes [152]. 3D shapes can also be reduced to two dimensional image representations, and image or 2D shape analysis methods can be used for further processing.

Vranic [153] creates two dimensional silhouettes of objects by projecting the axis aligned shapes on the sub coordinate frames. Those silhouettes are then described by a one dimensional function defined by distance from a center and then fast Fourier transform is applied. As a second descriptor, he uses six depth buffer images which are taken from faces of a cuboid that contains the shape. Each depth buffer is further transformed using two dimensional fast Fourier transform.

**Light field descriptors** [32] follow the idea that similar objects should look

similar from different view points and take silhouettes from camera positions defined at vertices of a dodecahedron. Then, the best similarity between two such descriptors is established while considering all possible rotations of one camera system relative to another and taking the minimum distance. For image metric Zernike moments and a Fourier descriptor of function, being a distance from center to a boundary, are used.

### 3.4.3 Harmonics on Arbitrary Shape

For most manifolds the analytic formulas for Laplace Beltrami eigensolutions are unknown but it is still possible to obtain their approximation numerically [123]. If as $\Omega$ we use the shape itself, either as a surface manifold or as an inner part of the shape with some boundary conditions, eigenvalues and eigenfunctions of Laplace Beltrami operator contain a lot of shape information. The eigenfunctions were already mentioned in the Section 3.1.5. Although there are different shapes with similar spectra, Reuter et al. [123] proposes to use eigenvalues as the **shape DNA** descriptor.

Rustamov [126] maps selected functions defined on a shape into template shapes, which can be further projected onto the eigenvalues of the Laplace Beltrami operator defined on the template shape. Many known methods, such as light field descriptors or spherical harmonics, can be described with the use of this framework, by specifying different functions on a shape, different mappings and different template shapes.

## 3.5 Segmentation

Retrieval of semantic information requires decomposition of shapes into meaningful parts [6]. Those parts are then treated as basic primitives that build the shape. The shape of each of those primitives and the relation between them can be further analyzed [129]. The meaning of those parts depends on the context which for example is different for mechanical parts than for articulated shapes.

Segmenting a shape is related to classification of local points of the shape. The main difference is that we also consider how different classes are positioned within a shape, as a segment is a connected set of local points of the same class. Segmenting algorithms aim to have a small number of parts with smooth or feature aligned boundaries. Shapira et al. [129] segments according to values of real functions defined on the surface mesh such as shape diameter. The

**Gaussian mixture model** is used to fit the k-Gaussians to a histogram of the shape diameter function. Then a probability of being one of those classes is assigned to each triangle of a mesh, according to fitted Gaussian values. The result is further regularized by using the graph cut algorithm, with energy containing a data term according to probabilities and a smoothness term according to dihedral angle between triangles and the length of the edge shared by the triangles.

Shamir et al. [128] also use a local function defined on the mesh and apply the **mean shift** algorithm. The algorithm finds maxima of the estimated probability density function on the feature space which is built from coordinates of local points and descriptors evaluated at those points. Points converging to the same value are considered as clusters. This method was imported from the image clustering and filtering technique. It requires some adjustments to work for meshes, in this case the mesh around the actual point is flattened and the algorithm works in euclidean space.

In work of Dey et al. [71] segments are constructed from a set of points that **flow** into the same critical points. The flow is defined along the gradient of the distance from the boundary.

Medial structures are used by Mortara et al. [101] in the plumber algorithm, which extracts tubular regions from the mesh. Medial loops are also used by Goes et al. [38] but distance between points is evaluated as a diffusion distance instead of geodesic distance. Here, a duality between medial loop and segment border is used while performing refinement operations.

Katz [75] segments shapes which consist of a core part and limbs. MDS scaling is performed based on geodesic distances between the vertices of a mesh. In such space, spherical mirroring is performed, and the convex hull of the mirrored part is computed. The core part of a mesh corresponds to the areas lying in the convex hull following the extraction of the limbs, which correspond to inner parts of the mirrored mesh.

Clustering with a **bottom up** approach starts with a huge amount of small regions, for example triangles in the case of a mesh representation. The cost of merging two neighbor areas is usually defined in some form of energy and, at each step, merging that has the smallest cost is performed. In the work of Attene et al. [8] clusters correspond to primitives, such as plane, sphere, cone or cylinder and are merged if the resulting area fits one of the primitives from the primitive set well. In similar fashion, Gal et al. [50] use the energy which is an error of fitting a quadric and Gelfand et al. [104] connect regions according to the slippage consistency: the number of eigenvalues of a slippage matrix and the signature of the slippage matrix has to be similar.

Huang et al. [64] define meaningful parts as those that tend to move rigidly and analyze vibration modes of the Hessian matrix of the deformation energy. Solomon et al. [138] segment according to intrinsic primitives and perform k-means clustering with distances based on the Killing vector field eigensolutions [14]. K-means clustering segmentation within the space of GPS coordinates, which are Laplace Beltrami eigenfunctions divided by the square root of the corresponding eigenvalue, was proposed by Rustamov et al. [125]. Xu et al. [161] suggest using intrinsic reflectional symmetry to help in the process of meaningful clustering of man-made objects.

Golovinsky et al. [59] computes the final segmentation along areas with the highest probability of having a cut, estimated from many random segmentations boundaries, based on the k-means clustering algorithm, the mean cuts algorithm and hierarchical clustering initialized with different parameters.

Kalogerakis et al. [74] use the conditional random field method with energy terms related both to geometry of parts and their pairwise relations. The objective function is learned from a collection of labeled training meshes. If two parts have the same label and are neighbors, they cannot be separated with that scheme.

## 3.6 Graphs and Skeletons

Shape skeletons are graphs whose elements are identified with different regions of a shape. Graph representations are connected to segmentations, since a graph representation can be obtained from the segmentation by connecting the neighboring segments with an edge [64]. Conversely, a shape can be partitioned according to the structure of its skeleton [91]. Matching and comparing general graphs is a big research topic in itself. This section aims at describing methods for extracting skeletons and matching techniques which use specific properties of skeleton representations.

**Shock graph** [135] is obtained by thinning a two dimensional shape and analyzing critical points of the thinning function. Sundar et al. [144] generalizes this idea to three dimensional shapes through volumetric thinning, with the use of additional information about the type of shock depending on the type of singularity that occurred when the graph was produced. A shock graph is a directed acyclic graph and direction is defined by the thinning direction, which makes the problem of matching much simpler than for general graphs. Partial matching is performed in coarse to fine hierarchical order. The measure of topological similarity of the subtrees, related to the number of eigenvalues of

the adjacency matrix, and measure of the local shape information are used to decide if two nodes are matched together or not. After matching, a goodness measure is produced which is combined from the number of nodes matched and accuracy of the match.

A similar approach to the shock graph can be seen in the work of Au et al. [9] where the mesh is contracted to a skeleton by applying constrained Laplacian smoothing.

A skeleton can also be obtained by analyzing the critical topological characteristics, such as critical points or curves, of a repulsive field over a discretization of the 3d object [37].

Sharf et al. [130] apply **deformable models**, a method which is typically used for surface reconstruction. A skeleton is created by following the centers of active fronts. The skeleton is simplified by removing arcs whose related front tension is smaller than a given threshold.

Given a manifold $\Omega$ and a Morse function $f : \Omega \to \mathbf{R}$, the **Reeb graph** [19] is a quotient space of $\Omega \times \mathbf{R}$ defined by the equivalence relation $(p, f(p)) \sim (q, f(q)) \Leftrightarrow f(p) = f(q)$ and $p$ and $q$ are in the same connected component of $f^{-1}(f(p))$. Reeb graphs trace the evolution of the level sets of functions defined on a manifold. Nodes of a Reeb graph correspond to the critical points of the functions, and edges represent connections between them.

Marini et al. [94] matches Reeb graphs of different shapes by a bottom up subgraph matching method, where initial mappings between nodes are expanded as long as possible. Expansion is always performed on a candidate pair which has the best heuristic similarity cost. Reeb graph is a directed graph and this property allows to reduce the complexity of expansion. Tierny et al. [150] use the fact that each of the Reeb graph edges has either tubular or disk like topology so they are mapped to a unit planar disk or a unit planar annulus. The stretch measures are used as descriptors for each part and a search for maximally similar subgraph is performed.

To establish correspondences and the similarity measure between two shapes Hilaga et al. [62] use the hierarchy of **multiresolutional Reeb graphs** calculated from the centricity function. At each level a Reeb graph of given resolution is computed. Reeb graph resolution is connected to a method where only selected level sets are examined, the ones with values placed equidistantly, and the distance defines the resolution [115]. Nodes of a multiresolutional Reeb graph correspond to the edges of the Reeb graph, and incident nodes are connected by an edge. To get a hierarchical sub graph, the resolution value is halved. In a result, each node from the coarser graph has corresponding children sub

nodes. Then, attributes are used to establish similarity between two nodes. A ratio of the area of the node to the whole object or length of the node to the whole object are used as attributes for matching in the finest resolution. For the coarser resolution, the attribute of a node is defined as the sum of the attributes in the subnodes. Correspondances are established using the coarse to fine strategy in such a way that nodes in different 'branches' of a Reeb graph should not be matched. Similarity is calculated as the sum of similarities between matched nodes. In an augmented version of the algorithm [151], the matching cost is enriched with distances between different geometrical attributes such as the spherical coordinates of the node centers, statistic measure of the extent, relative volume, locally estimated shape index etc.

**Medial axis transform** [22] of a surface F can be defined as a set of medial balls, the maximal empty balls that do not contain any point of F and are not contained in any other such balls [3]. For two dimensional shapes medial axes have a skeletal structure. For the 3D case, medial axes contain surfaces. There are methods for further extraction of medial axes from those surfaces which result in a skeleton. For example, Dey and Sun [41] extract such skeletons by using critical points of the geodesic function, which is defined for a given center of a ball as the geodesic distance between a pair of points touching this ball.

Medial axes representation is noisy with respect to shape perturbations. Small noise at a boundary might cause addition of lot of new branches. Filtrations are used to reduce the medial axis transform noise. Such a filtration is a subset of the medial axis with a set of points for which any ball containing the set of closest points on the boundary has radius at least $\lambda$ [31]. However, by using $\lambda$-filtrations, thin but long elements might be lost. Scale axis transform [56] is a version which scales the radii of the medial balls by a factor s. That way, some balls are no longer maximal as they are contained withing other balls. After performing medial axis transform on the surface defined by scaled balls, radii are scaled back to the original size.

## 3.7 Persistence Diagrams

**Size functions** [47] analyze variations of the number of connected components of the lower level sets of a manifold $S$ with respect to a continuous function $f$. A size function has two parameters $x$ and $y > x$. It counts the number of connected components of $S_y$ which contain at least one point of $S_x$, where $S_x = \{p \in S : f(p) < x\}$. The functions can be represented as collection of cornerpoints and cornerlines. Size functions can be compared by computing the Hausdorff distance between the corner element sets of two shapes, with allowing

the points to be matched with the diagonal $x = y$.

For 3D representation size function very often becomes trivial, so a setup requiring two functions was proposed [18]. First function extracts the skeletal graph according to level sets, and the second function measures geometric properties at a given node of the graph, and this function defines the filtration.

Persistent homology [44] is a more general framework for measuring topological features of manifolds or, in a discrete case, simplicial complexes. It analyzes all kinds of topological changes that can happen while incrementally growing the space. The growth of that space is defined by ordering of the added elements or by some time-parametrized filtration scheme. The events of topological change can be paired as births and deaths of different homology classes and displayed as points on the persistence diagram. The difference between death and birth time define the persistence of a given feature, which on a persistence diagram, can be measured as the length from the diagonal of a persistence diagram. The distances between two persistence diagrams can be computed as the Hausdorff distance by matching the points of the diagram, which is the assignment problem, and can be solved for example by the Hungarian algorithm. Note that persistence was used by Dey et al. [40] in order to find the most significant maxima of the auto diffusion function.

An interesting example of using persistence homology is a **barcode** descriptor which does not act on a shape directly but in the space of a tangent shape. Barcodes are defined [27] as the set of intervals which describe the persistence of filtration of tangent complexes of the shape. Those tangent complexes are filtered with respect to the osculating circle radii. The filtration with parameter $\delta$ removes those directions in the tangent space that have the tangent curvature greater than $\delta$.

## 3.8 Direct Comparison via Matching

Evaluating distances between shapes is possible without using intermediate representation in the form of a shape descriptor.

If two shapes are aligned in the same coordinate space a Hausdorff distance between them can be computed as the maximum, over all points from the two shapes, of distances from a given point to a closest point from the second shape. Some other distance formulations are also possible. For example, if the volume enclosed by given shapes can be measured, we can use Jaccard distance [67], which measures the volume of XOR operation evaluated on both shapes, divided

by the SUM.

Usually we do not have perfectly aligned shapes. A Gromov Hausdorff [100] distance is defined as minimum Hausdorff distance among all possible embeddings of shapes in the space. There is also an alternative formulation which does not require common embedding space but correspondences between points of two sets need to be established.

From this example we can see that shape matching methods are also a way of establishing distance based shape descriptors. Note also that many matching methods minimize some functional and such functional can be used as a distance measure. For example Zhang et al. [164] uses as a cost measure the energy of deforming a mesh according to correspondence established points.

Matching curves (see also Section 2.3) is very often done by applying the dynamic time warping algorithms [45] and minimized objective function can be further used as a distance measure between those curves. For surfaces, the use of dynamic time warping becomes NP-hard. Usually 3D methods work on coarse correspondences of feature points [164], also methods based on exploration of Möbius [81] space, or isometric embedding based on heat kernel [113] can be applied for almost isometric shapes.

## 3.9 Combining Different Measures of Dissimilarity

Very often one type of descriptor is not enough to capture all properties that we need. In the retrieval framework, combined descriptors usually perform much better than base descriptors [58]. In order to combine different descriptors a common representation is needed. Although there are many various types of descriptors, most of them have a way of measuring dissimilarities.

Giorgi et al. [57] uses relevance feedback which is given as a threshold number assigned to a specific *feedback* shape in the database. Combining different dissimilarities is done by taking the maximum one. The feedback from the user is taken into account by scaling down measures in a way that $d_i(qery, feedback)$ is no greater then threshold. If a measure is within a given threshold it remains unscaled.

Bustos et al. [26] combines the descriptors by summing the normalized distances with weights computed according to impurity measure function. Impurity is related to fractions of objects of the same class at k first positions within the

retrieval list. This is based on a hypothesis that a well suited descriptor will retrieve objects from the training set that belong to the same class. The framework requires a training dataset with labeled classes. Note also that in this scheme weighting depends on the actual query shape.

Ceri et al. [29] propose the use of **a Contrario model** which is a statistical method of combining information provided by different dissimilarity measures $d_{i=1..N}$. The need for combining descriptors is in this case motivated by the fact that size functions are used as descriptors and different shapes might have similar size functions. At the time of evaluating how similar $S' \in B$ is to a query shape $S$, all distances from $S$ to $B$ need to be known. The scheme is based on probabilities and each measure is assumed to be independent, so all terms related to measures are combined by multiplication. Each such term is equal to the ratio of shapes in $B$ which are not more distant to $S$ than $S'$. This term multiplied by the number of shapes in $B$ is known as the **number of false alarms** and is used as a dissimilarity measure for the retrieval problem.

Chapter 4

# Contributions

The goal of this project was to find meaningful ways of describing shapes that agree with the ways that human mind analyzes them. Especially, we were motivated by the task of capturing the features related to style.

The contribution of this thesis to the area of shape description methods is twofold.

The first is to propose a descriptor, which is related to the structure of the shape, namely the auto diffusion function. With this descriptor we also show a method of finding meaningful shape subparts, and a very compact and informative way of describing them.

The second contribution rather than designing new descriptors contains methods for dealing with existing shape descriptors in the context of style and function related problems. We assume in this work that any descriptor which produces a measure of dissimilarity between two shapes can be used, which makes our framework very general. We introduce a method on how to decouple the information about style and about function if they are contained within the same dissimilarity measure. We also give a way of assessing how style oriented a given descriptor is, within the context of a specific training dataset. We propose a consistency measure which was mostly motivated by the problem of replacing a missing piece of an exemplar style set, by selecting the most similar style from

the repository dataset which contains style and function labeled shapes.

Chapter 5

# Shape Analysis Using the Auto Diffusion Function

Katarzyna Gębal
Jakob Andreas Bærentzen
Henrik Aanæs
Rasmus Larsen

Technical University of Denmark

Published in *Comput. Graph. Forum, 28(5):1405-1413, 2009*

Scalar functions defined on manifold triangle meshes is a starting point for many geometry processing algorithms such as mesh parametrization, skeletonization, and segmentation. In this paper, we propose the Auto Diffusion Function (ADF) which is a linear combination of the eigenfunctions of the Laplace-Beltrami operator in a way that has a simple physical interpretation. The ADF of a given 3D object has a number of further desirable properties: Its extrema are generally at the tips of features of a given object, its gradients and level sets follow or encircle features, respectively, it is controlled by a single parameter which can be interpreted as feature scale, and, finally, the ADF is invariant to rigid and isometric deformations.

We describe the ADF and its properties in detail and compare it to other choices of scalar functions on manifolds. As an example of an application, we present a pose invariant, hierarchical skeletonization and segmentation algorithm which makes direct use of the ADF.

## 5.1 Introduction

Many algorithms for processing or analysis of manifold shapes are aided by scalar functions defined on the surface. It turns out that functions which are smooth and aligned with the shape are excellent tools for parametrization and extraction of topological information. In particular, harmonic functions, piecewise harmonic functions or eigenfunctions of the Laplace Beltrami operator have been used a great deal. However, in order to define a harmonic function on a mesh, its boundary conditions need to be specified (sources and sinks), and while the eigenfunctions of the Laplace Beltrami operator carry a great deal of shape information, it is often hard to find out precisely what each eigenfunction represents. In particular, the picture gets more complicated when eigenfunctions corresponding to eigenvectors larger than the first non-zero eigenvalue are used.

Our goal is to find a smooth scalar function which attains its maxima on the tips of features (tips of fingers, noses, tentacles, hoofs, tails, tops of heads, etc.). Since the notion of a feature is scale dependent, we observe that the function must be parameterized by a scale parameter, but it would defeat the purpose if the user had to specify other information (e.g. feature points). Furthermore, we desire that the function be invariant to any rigid transformation, scaling and to

isometric deformations of the shape. We believe such a function would be highly useful for a number of tasks such as feature point detection, skeletonization, segmentation, and parametrization. However, in this paper, we focus on showing the usefulness in relation to skeletonization and segmentation.

### 5.1.1 Contributions

Our main contributions are to describe the *auto diffusion function* (ADF) which is a tool for shape analysis and to investigate its applications to geometry processing.

Assume we assign some quantity, like for example heat or dye, subject to diffusion to a single point on a manifold shape. For a point on this shape, $x$, and a scale (or time) parameter $t$, the $ADF_t(x)$ is the fraction of the quantity that remains at $x$ after time $t$. Intuitively, it is fairly clear that the quantity that remains will be bigger on or near features since for a family of metric disks centered at $x$, the ratio of area of metric disk to its perimeter will be bigger for a feature point than for a point on a flat part of the shape. Thus, more of the initial quantity escapes in the flat case. Detailed description of ADF comes in Sections 3 and 4, see also [16, 30, 124].

In Figure 5.1 we show a complex shape with the ADF compared to the Fiedler vector (first eigenfunction corresponding to a non-zero eigenvalue). It is clear that all the tentacles of the shape are encircled by iso-curves of the ADF while that is not true of the Fiedler vector. We could have used a harmonic function, but compared to harmonic functions, the main advantage of the ADF is that it does not require the setting of any boundary conditions on feature points. Instead, its extrema prove to be the natural feature points.

We also propose an algorithm for skeletonization and segmentation based on the well known method of Reeb graphs applied to our ADF. Our contribution pertains to the fact that a single branch point often turns into multiple bifurcations on a Reeb graph, and each bifurcation tends to lie on or near the surface. We address this issue by averaging points on the Reeb graph within some small tolerance of a critical value. The final skeleton is combined from several Reeb Skeletons, which are extracted using the ADF function evaluated at different $t$-values.

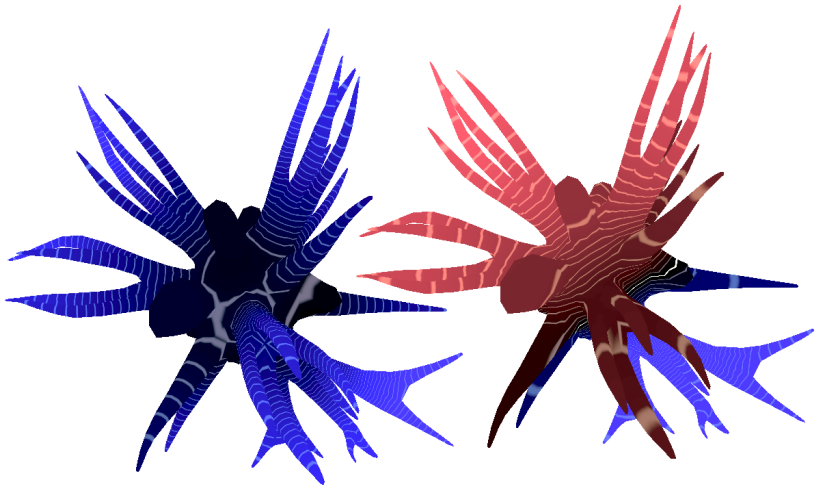


Figure 5.1: ADF and Fielder functions for the tentacles shape. The rings are iso-curves and positive values map to intensity of blue while negative values map to intensity of red. Notice how the level sets of the ADF encircle the tentacles of the shape much more consistently than the Fiedler vector.

## 5.2 Related Work

### 5.2.1 Functions on Surfaces

Ni et al. [107] find a harmonic function by solving the Laplace equation with selected vertices as the constraints. This yields a smooth function that has maxima only at the given points, however it is necessary to provide the extrema in advance. Moreover, the values of the extrema need to be carefully assigned depending on the lengths of the protrusions. Otherwise, the saddle points will be misplaced from the natural branching area. The fast computation of harmonic functions in [160] allows for an interactive approach where the user defines all of the constraint points and obtains a new harmonic function immediately. Dong et al. [42] propose to use curve constraints as the boundary conditions. These constraints are imposed by requiring that all vertices which belong to the curve need to have the same value of the function. There exist some heuristic algorithms aimed at finding good constraints for harmonic functions [42,70,140].

Given a single user provided source point, Aujay et al. [10] find the sinks as the clustered maxima of the function of shortest distance to the source. They create

harmonic functions which have boundary conditions defined at the sources and sinks. Tierny [148] finds feature points automatically as the intersection of two sets of points which, in turn, are the extrema of the geodesic distance to one of a pair of diametrically opposite points. The scalar function is then the geodesic distance to the nearest feature point. In [149] the geodesic distance is modified by making areas separated by concavities, more distant. For more details see Section 4.1. In general, methods based on geodesic distance are not always the best choice because they are very sensitive to the small topology changes as mentioned in [125].

When using the ADF or eigenvectors of the LBO it is not necessary to provide extrema because they are defined by the function itself. The existing approaches based on eigensolutions of Laplace Beltrami operator usually involve choosing one of the eigenfunctions. Some of the papers [115, 122] suggest exactly that but without specifying which one to choose. [131] takes the first nontrivial eigenfunction and creates the skeleton based on that. However, this approach may overlook many shape details especially when the nontrivial eigenvalues are almost the same. [43] picks one of the Laplacian eigenfunctions according to a given number of critical points that the function produces. But in this way one imposes the complexity. For two shapes of very different complexity (say the tentacles shape in Figure 5.1) this would lead to very different levels of detail which is not likely to be desirable.

### 5.2.2 Skeletonization and Segmentation

Many methods for producing a skeletonization require either a volumetric representation or a Voronoi diagram. Many of these are discussed in [9]. Most of them create a lot of tiny branches and are sensitive to noise, others have a big computational cost. Below, we restrict the discussion to those that directly work on the surface mesh since that is most pertinent to this paper. [115, 131] create a Reeb graph directly from the choosen LBO eigenfunctions and make a segmentation and shape skeleton from this graph. Au et al. [9] get the skeleton by shrinking the mesh using Laplacian smoothing. However it does not work for coarse meshes, and it does not seem possible to specify the desired level of detail.

Many segmentation algorithms are based on geometric properties of the parts, for example their convexity or the local curvatures. As we require pose invariance, we refer only to methods which meet this requirement. Katz et al [76] use geodesic distance and angular distance as a metric for the k-means clustering method. The number of clusters is obtained by taking the k that has the biggest derivative of minimal distance of the k-th added representative from other repre-

sentatives. In a later paper [75], Katz et al. obtain the pose invariance by using multi-dimensional scaling (MDS) to make the geodesic distances similar to the Euclidean distances in 3D space. The points that reside on the convex hull of this representation are chosen as the feature points, then spherical mirroring is used to extract the core component and the feature components. In the end, the boundaries of the segments are refined by finding the optimal cut. This method seems to work only with shapes that have a distinguishable core part.

Tierny et al [149] use an approach based on Reeb graphs evaluated on geodesic and curvature based functions. The segmentation process is not only aided by the topological information but it is also enhanced by the placement of the constrictions – the curves at the bottlenecks of the shape which are good candidates for the boundaries of the segments. Some heuristic methods like the identification of the core part of the object is conducted in order to remove some of the constrictions and produce the final shape.

Spectral methods [165], especially based on eigendecomposition of the Laplace Beltrami operator have pose invariance induced by the properties of the operator. Rustamov [125] proposes k-means clustering based on inner products of points in GPS coordinates. The GPS coordinates of a vertex is the vector of values of the LBO eigenfunctions at that vertex where each value is divided by the square root of the corresponding eigenvalue. Reuter et al. [122] mention segmentation based on the nodal domains of some of the eigenfunctions. In both of those methods, the user needs to provide information. Liu et al. [90] perform 2D contour analysis of the shape in the space of the first two eigenfunctions. The problem is that these may fail to capture even basic shape properties. In [64] modal analysis of the Hessian of the deformation energy is used to find the typical low energy deformations. Decomposition of the shape into parts is done by finding the optimal approximation of those deformations by deformations which are rigid for each segment.

[38] proposed to use the diffusion distance between points to make a shape segmentation. Like the ADF, the diffusion distance is defined in terms of the Gaussian kernel on the mesh, but the functions are otherwise very different: The diffusion distance only makes sense for two points since the diffusion distance from a point to itself is identically zero.

## 5.3 Theoretical Background

### 5.3.1 The Laplace-Beltrami Eigensolutions

The Laplace-Beltrami operator (LBO) is defined on a Riemannian manifold $\Omega$ as the divergence of the gradient of a scalar function $f : \Omega \to \mathbf{R}$.

$$\Delta f = \nabla \cdot (\nabla f)$$

The LBO can be found in equations describing physical phenomena such as wave propagation and heat distribution. This operator has also been used extensively by the computer graphics community in the last few years for many purposes such as: mesh smoothing [105], parameterization [103, 118], editing [139], morphing and deformation [2]. The decomposition [86] of the LBO into eigenvalues $(0 \leq \lambda_0 \leq \lambda_1 \leq \dots)$ and eigenfunctions $(\phi_0, \phi_1, \dots)$, which can be expressed as the solution of the Helmholtz equation $\Delta f + \lambda f = 0$, has also lately been explored in a lot of applications for geometry processing including shape identification [123], classification [125], segmentation [90], registration [95] or identifying the shape symmetries [114].

The eigenfunctions corresponding to the first few eigenvalues align surprisingly well with the protrusions and features of the object. This phenomenon can be explained by investigating the Rayleigh Quotient method which is used for calculating the eigensolutions to the symmetric operator, which in the case of the Helmholtz equation is equal to $-\Delta$. The problem is stated as a minimization problem:

$$\phi_i = \underset{\substack{0 \neq f_i \in C_0^2(\Omega) \\ \langle \phi_{j \in \{0..i-1\}}, f_i \rangle = 0}}{\arg\min} \frac{\langle f_i, -\Delta f_i \rangle}{\langle f_i, f_i \rangle}$$

The minimal value is equal to the corresponding eigenvalue $\lambda_i$. By applying the fact [30] that for the compact manifold the divergence and minus gradient are formal adjoint operators $\langle \nabla f, \mathbf{X} \rangle_v = -\langle f, \nabla \cdot \mathbf{X} \rangle$ to the vector field $\mathbf{X}$ equal to $\nabla f$ [11], we transform this problem into the mimimization of the $\langle \nabla f_i, \nabla f_i \rangle_v$, which is equivalent to the Dirichlet Energy minimalization problem with the constraints $\langle f_i, f_j \rangle = \delta_{ij}$, where $\delta_{ij}$ is here the Kronecker delta. Note that we use here two different inner products: the first one $\langle f, g \rangle = \int_\Omega f(x) g(x) dx$ is the inner product of two scalar functions over our manifold $\Omega$, and the second one $\langle \mathbf{X}, \mathbf{Y} \rangle_v = \int_\Omega \langle \mathbf{X}(x), \mathbf{Y}(x) \rangle_\Omega dx$ is the inner product of two vector fields, where $\langle , \rangle_\Omega$ is the inner product defined by the structure of our Riemmanian manifold $\Omega$ [124].

Put more loosely, the eigenfunctions are mutually orthogonal, have as small as possible gradients everywhere while $\langle \phi_i, \phi_i \rangle = 1$. For a closed surface, a constant

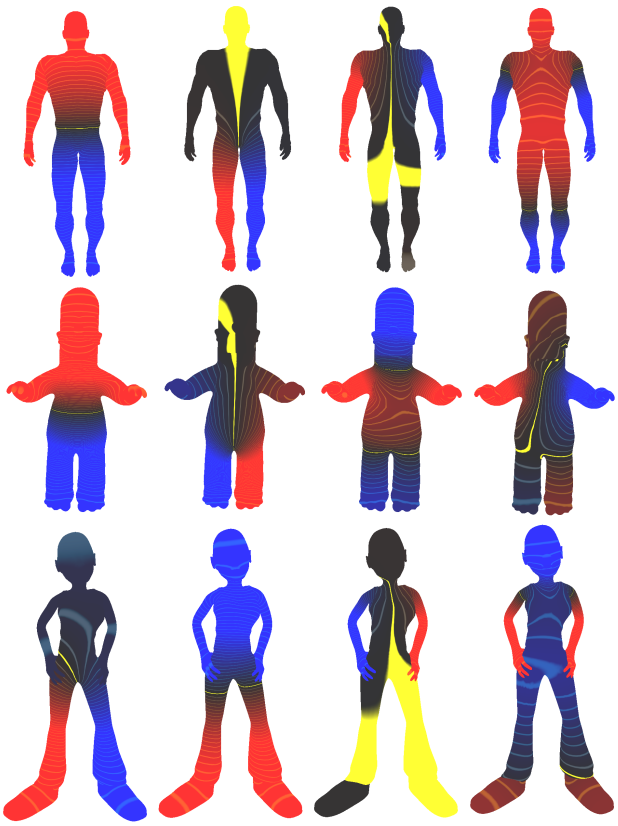


Figure 5.2: First four nontrivial eigenfunctions of the LBO for three different human like shapes. Red color represent negative and blue positive values, yellow shows area close to the nodal set. The order of the eigenfunctions depends on the proportions of the specific parts. Note that the nodal sets do not always cut the object symmetrically.

function will suffice as $\phi_0$. However, $\phi_1$, the Fiedler vector, must be orthogonal to $\phi_0$ and consequently it is positive on half the shape and negative on the other half. The requirement that the gradients should be as small as possible translates into the well known fact that the direction of change of the Fiedler vector naturally follows the shape [86] as illustrated in Figure 5.2 left column.

There is an analogy between the eigenfunctions of the LBO and harmonic functions mentioned in Section 2.1. Both minimize the Dirichlet energy, but the eigenfunctions are constrained by the orthogonality requirement rather than boundary conditions.

While using the LBO eigenfunctions, there are some important issues that need to be remembered: The signs of eigenvectors are undefined, two eigenvectors may be swapped, nodal sets can be unstable due to small metric changes. All of these are addressed by the Auto Diffusion Function presented below. In our implementation $\Omega$ is represented as a triangular mesh. We use the cotan weights [39, 118] to calculate the entries for the LBO and we solve a generalized eigenproblem in a way similar to the one used by Rustamov [125]. Because we are interested mostly in the first few hundreds of eigenvalues we use the sparse solver ARPACK together with SuperLU.

### 5.3.2 The Diffusion Kernel

The diffusion kernel $K(x, y, t)$, or heat kernel, is a fundamental solution (Green's function) to the heat equation:

$$(\Delta_x + \partial_t)u(x, t) = 0$$

where $\Delta_x$ denotes the Laplace Beltrami operator acting on the spatial variable $x$ where $t \in [0, \infty)$ is the time variable. It solves the equation with the initial condition $u(0, x) = \delta(y)$, where $\delta(y)$ is Dirac delta function at the position $y$, which means that all heat is initially concentrated in one point at the position of $y$. The general solution can be obtained by convolution of the heat kernel with the initial condition $g(x) = u(0, x)$. The heat kernel can be expressed [124, page 32] in the terms of LBO eigensolutions as:

$$K(x, y, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t}\phi_i(x)\phi_i(y)$$

## 5.4 The Auto Diffusion Function and its Interpretation

If we inject a unit amount of some quantity like dye at a point $x$ (this corresponds to the initial Dirac delta function), the Auto Diffusion Function indicates how much of the dye that remains after time $t$. So the ADF describes the diffusion from the point to itself and can be written as:

$$K(x, x, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t}\phi_i^2(x)$$

In this way, we obtain a function which is only dependent on the eigenvectors and eigenvalues of LBO, which, in turn, depend solely on the first fundamental

form G, and is therefore independent of isometric deformations, translations, and rotations. To make the function scale invariant we can simply divide the exponential by the second eigenvalue [121]. So we define the Auto Diffusion Function as:

$$ADF_t(x) = K(x, x, \frac{t}{\lambda_1}) = \sum_{i=0}^{\infty} e^{-t\frac{\lambda_i}{\lambda_1}} \phi_i^2(x) \tag{5.1}$$

As the equation shows, the ADF is simply a linear combination of squared LBO eigenfunctions. If the parameter $t$ is large, the eigenfunctions corresponding to big eigenvalues count less, and then only the main features 'detected' by the smaller eigenvalues influence the ADF. As we decrease $t$, more features can be seen (cf. Figure 5.3).

The physical interpretation of the ADF is given by the diffusion process: At the tips of protrusions, less of the dye will escape than from flat areas. But if $t$ is big enough, and the protrusions are small, there will be enough time for the dye to spread evenly to neighboring areas. On the other hand, for small $t$ there must be a connection between local Gaussian curvature and (5.1). This connection can be made explicit by observing the Minakshisundaram-Pleijel expansion of the heat kernel. For sufficiently close $x$ and $y$

$$K(x, y, t) = (4\pi t)^{-n/2} \sum_{k=0}^{\infty} u_k(x, y)\, t^k$$

where $n$ is the dimension of the Riemannian manifold and $u_k(x, y)$ are recursively defined. If $y = x$ then $u_0(x, x) = 1$ and $u_1(x, x) = \frac{1}{6}S(x)$, where $S(x)$ is the scalar curvature which is the Gaussian curvature in the case of a 2-manifold. Therefore, we have

$$ADF_{t\lambda_1}(x) = (4\pi)^{-1} \left( \frac{1}{t} + \frac{S(x)}{6} \right) + O(t) \tag{5.2}$$

However, (5.2) is not a viable alternative to (5.1) since the truncated terms are insignificant only for small $t$. Conversely, (5.1) is faster to compute for bigger $t$ since fewer high frequency eigenfunctions are needed.

In practice, we do not include the first eigenfunction since it is constant, and we only add eigenvectors corresponding to eigenvalues that fulfill $e^{-t\frac{\lambda_i}{\lambda_1}} < \delta$, where the threshold is $\delta = 0.01$.

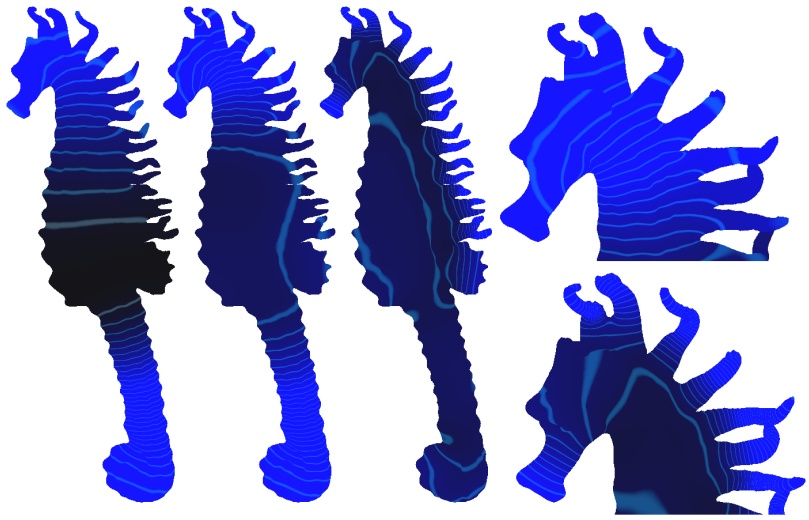


Figure 5.3: ADF evaluated with $t$ equal to respectively $2, \frac{1}{2}$, and $\frac{1}{32}$ (left to right). In the magnified images t is $\frac{1}{2}$ (top) and $\frac{1}{32}$ (bottom). The lines show the isocontours of the function; brighter color indicates bigger value of the ADF.

### 5.4.1 Comparison to the Existing Functions

In this section, we compare the ADF to some other functions as illustrated in Figure 5.4. We concentrate on those functions that can be specified with no direct annotation of feature points and are pose invariant.

From our point of view, using the geodesic distance to the closest feature entails two problems. First, the function has first order discontinuities since it involves taking the minimum of several geodesic distances. The second issue occurs when some feature point is located at a much smaller protrusion than a neighboring protrusion. Then, most of the surface at the 'parent area' of the protrusion is geodesically closest to the feature point of the smallest protrusion. This can lead to a situation where the function at the 'parent area' is mostly defined as the distance to the point corresponding to the smallest detail. Observe what happens with the isocurves close to the little finger on the last two hands on the image 5.4 that represent Tierny's geodesic functions. Even with the geodesic distance enhanced by the curvature information the problem is still present.

The Fiedler vector aligns well with the overall shape, but some parts of the mesh close to the zero level set are poorly aligned with the protrusions. One general problem with using the Fiedler vector is that for highly symmetric objects, the

smallest eigenvalues will be nearly identical, and the first few eigenvectors align with directions of roughly the same significance. The advantage of the ADF is that the weights of the eigenfunctions depend on the eigenvalues, so all of the first few eigenvectors may contribute. As noted previously, for small values of $t$, the ADF starts to resemble Gaussian curvature.

Figure 5.4: Different functions calculated on the human hand. The first two top represent ADF for t=1 and t=0.1, the bottom two are the Fiedler vector and highly smoothed Gaussian curvature, and the last two are geodesic based (pure geodesic and enhanced by the curvature) from [148] (pictures used with the author's permission).

## 5.5 Feature Based Skeletonization and Segmentation

### 5.5.1 Reeb Graphs

Calculation of the Reeb graph is a natural application of functions defined over a manifold. Given a manifold $\Omega$ and a function $f : \Omega \to \mathbf{R}$, $p$ is a critical point of $f$ if $\nabla(f(p)) = 0$. $f$ is a Morse function when for each critical point $p$, the Hessian martix $H(f(p))$ of the second order derivatives of $f$ is non-singular, which means that critical points of $f$ are non-degenarate.

The Reeb graph [7] of a Morse function $f$ on a manifold $\Omega$ traces the evolution of the level sets of the function on the manifold and is the quotient space of $\Omega \times \mathbf{R}$ defined by the equivalence relation $(p, f(p)) \sim (q, f(q)) \Leftrightarrow f(p) = f(q)$ and $p$ and $q$ are in the same connected component of $f^{-1}(f(p))$.

The nodes of the Reeb graph correspond to the critical points of the function f and the arcs, which we call the Reeb edges, represent the connections between them. The Reeb graph itself is a topological construct. However, if the manifold $\Omega$ is embeded in Euclidean space, the point corresponding to each connected component of the level set function can be positioned at the center of mass of the corresponding component of the level set curve. This we denote the Reeb skeleton.

In the discrete case where the manifold is a triangular mesh, the function $f$ is defined over the vertices. The points can be classified as regular or critical according only to the function values of the vertices that belong to the one ring of the vertex. The critical points are sinks if they are at the maxima, sources if there are at minima and saddles otherwise. The full description of the point classification according to the one-ring of the vertex can be found in [36]. Our Reeb graph computation is done using a sweep algorithm that allows us to position the skeleton points in space. The algorithm is similar to the one described in [36]. If the positioning of the vertices is not needed, a faster algorithm, that analyses the iso-contours only at the critical points [115], can be used.

### 5.5.2 Algorithm Outline

Reeb skeletons tend to look nicest at points which are not close to branch points. Branch points have a tendency to be either on or very close to the

surface rather than inside the shape where one would normally place a branch point. Also, where a designer would typically have placed a single branch point, Reeb skeletons tend to have multiple bifurcations. We address both problems by defining a small interval around the critical function value corresponding to a branch point. All points on the Reeb graph corresponding to values within this interval are averaged into a single point which we denote a *joint*. The remainders of the edges of the Reeb graph are denoted *bones*.

The method is somewhat similar to the extended Reeb graph method [7, 21] which slices the shape into parts that have function values between probing points according to a given frequency. Both methods work on critical areas instead of the critical points which helps avoid degenerate situations. Critical areas for saddle points correspond to our *joint* areas, but in our case the critical points are placed in the middle of the sliced interval, and neighboring critical areas that have small differences in function values (for a given $t$ parameter of the ADF function) are merged together in order to avoid topological noise.

The skeleton is combined from several Reeb skeletons which are extracted with $ADF_t$ evaluated with different $t$-values. Usually, we refine a skeleton by computing a new one at $t \leftarrow \frac{t}{2}$ and then grafting details from the new fine skeleton onto the coarse skeleton. We could also just compute a family of skeletons using different $t$ values for the ADF function, but this would make it hard to absolutely guarantee that we obtain a hierarchy, i.e. that there are no details in the coarse skeletons which later disappear in a fine skeleton.

### 5.5.3 Extracting *Joints* and *Bones* from the Reeb Skeleton

We denote as the *Reeb edge area* $E$ all the points that are transfered into the same arc of the Reeb graph. Each Reeb edge has two critical points at the endpoints, the smaller one $p_s^E$ and the bigger one $p_e^E$. So for each point $q$ that belongs to the given edge area $ADF_t(p_s^E) \leq ADF_t(q) \leq ADF_t(p_e^E)$. From each edge area, two *joint* areas are extracted at both ends and they are defined as:

$$J_s = \{q \in E : ADF_t(p_s^E) + \varepsilon \geq ADF_t(q)\}$$
$$J_e = \{q \in E : ADF_t(p_e^E) - \varepsilon \leq ADF_t(q)\}$$

The threshold epsilon used in those equation is:

$$\varepsilon = \kappa\mu = \frac{\kappa}{N} \sum_{i=0}^{\infty} e^{-t\frac{\lambda_i}{\lambda_1}}$$

where $N$ is the number of vertices in the mesh and $\kappa$ is a user defined threshold, we use $\kappa = 0.05$, which works fine, but can be changed according to the user preferences. $\mu$ is the hypothetical value of the $ADF_t$ if all of the eigenfunctions were constant. Such an objective formulation helps in an intuitive threshold manipulation as $\kappa$ can be the same for ADFs with different $t$ values and different shapes.

Neighbouring *joint* areas are glued together into one *joint* area. If the edge is small $|ADF_t(p_s^E) - ADF_t(p_e^E)| < 2\varepsilon$ then the whole edge area is added into the *joint* area and the two critical points that are the endpoints of this edge are merged together. If a *joint* contains sinks, this means that some geometric details, such as fingers, are not fully captured, because of the threshold. Such joints are indicated as *improvement*. Edge areas from the edge not contained in the *joint* areas are called *bones*.

An edge area $E$ which has a sink at the end $p_e^E$ is treated in a special way since only the $p_s^E$ is a joint endpoint. However, if the $|ADF_t(p_s^E) - ADF_t(p_e^E)| < 3\varepsilon$ the whole edge is added to the edge area.

All the sink edges are integrated into the joint. This kind of edge is defined recursively as an edge that has no brother edges and its parent is a sink edge or it has no parent. For the parent $P$ and the brother $B$ the critical points are shared in a way that: $p_e^P = p_s^E$ and $p_s^B = p_s^E$.

In this way we get a graph structure with the *bones* being the edges of the graph and the *joints* being vertices. An edge is incident with a vertex if its Reeb edge area have *joint* areas belonging to the *joint* areas corresponding to this vertex.

### 5.5.4 Integrating *Joints* and *Bones* from Different Reeb Graphs

As described, the algorithm produces a skeletal structure at a single LOD, but it is possible to go to a finer level of detail by refining the *improvement joints*: The structures from two Reeb graphs $G_1$, $G_2$ corresponding to different parameters $t_1 > t_2$ are merged together in the following way. If there is a new *bone* from $G_2$ that has both endpoints at the same *improvement joint* area from $G_1$ then this edge is grafted onto that *improvement joint*. The *bone* area is taken off from the *joint*. If the *bone* disconnects the joint then new joints are created and the edges are carefully connected to their incident joints. Refinement is illustrated in Figure 5.5.

The operation of merging new Reeb graphs with the existing structure can be

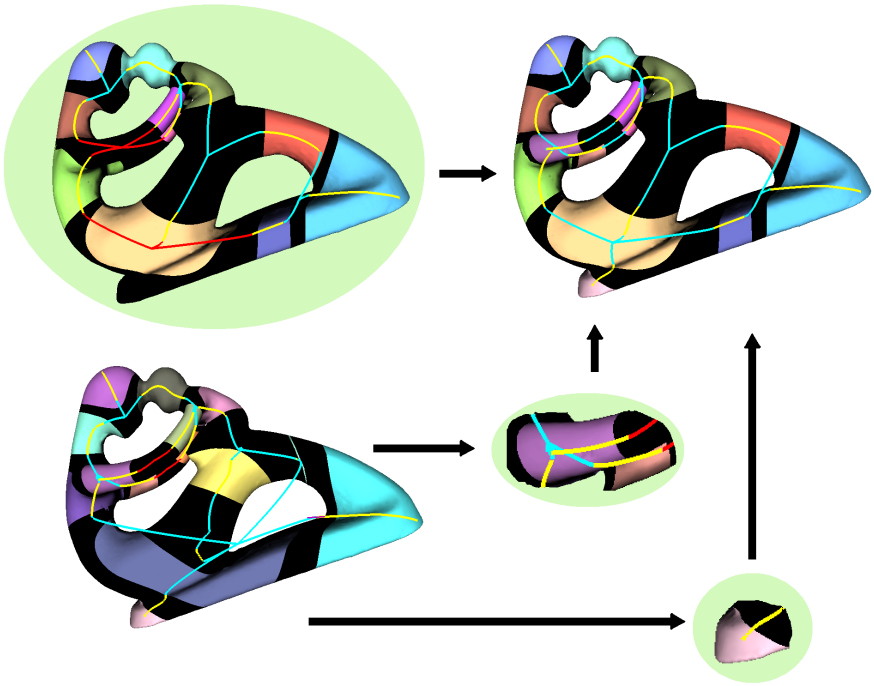


Figure 5.5: Ilustration of the process of merging two different graphs. Joint areas on the mesh are marked black. Improvement joints are red, regular joints blue, and bones yellow. The top left coarse mesh is connected with the bottom left one. From the detailed mesh only those edges are taken which belong to the improvement joints area (middle and bottom right).

repeated with gradually smaller values of $t$, until we have no *improvement joints* or up to the desired level of detail.

### 5.5.5 Defining the Final Skeleton and Segmentation

The geometry of the skeleton is computed as follows: For each *joint* we calculate the center of mass of the *joint* area, which we call the *joint* center, and then we connect to it the incident *bones*. If we are at a one-parent-many-children *joint* then it looks more pleasant if the *joint* center is moved towards the parent, so in that case we calculate the *joint* center as the center of parent *joint* area. The parts of the skeleton corresponding to the *bones* are the centers of the level sets. However, the bones need to be connected to the joint: The parts corresponding

Figure 5.6: Segmented poses of the Armadillo. On the top, the segmentations have been created from a coarse skeleton ($t = 1$,$\kappa = 0.04$) and below we see the result after refining the improvement joints ($t = 0.25$). Note that there is some noise added to the last pose of the Armadillo to demonstrate insensitivity to noise.

to the *joint* areas of the edges are a weighted average of the centers of level sets and the *joint* center. The closer we are to the critical point the more position of the *joint* center counts so finally for the critical point the position is equal to the position of the *joint* center.

When doing segmentation, we integrate some of the *joint* areas with the bone areas. If the *joint* connects only two *bones* then there is no branching at this

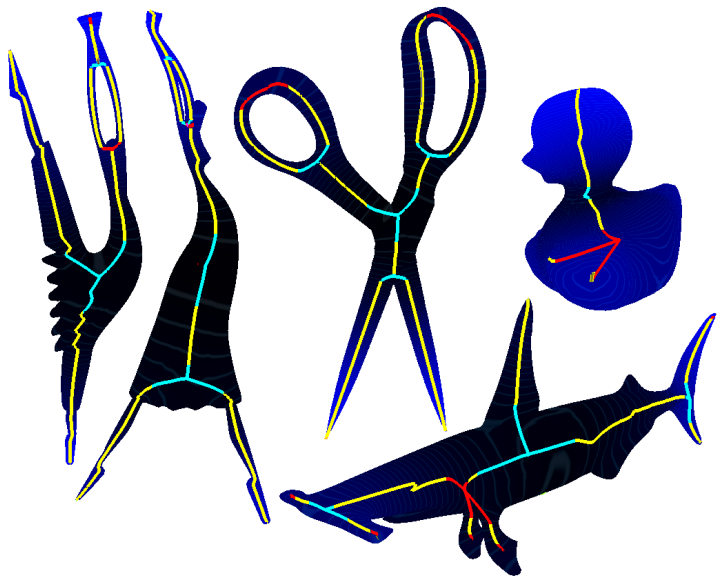


Figure 5.7: Skeletons produced for different meshes with $t = 0.5$ and $\kappa = 0.04$ drawn on top of the ADF for the mesh. Improvement joints are red, regular joints blue, and bones yellow.

level of detail. In that case those two *bone* areas and the *joint* area are connected together into one segment. The *joint* area is also integrated into a *bone* area if it has only one *bone* connection. In the one-parent-many-children case we connect the *joint* node to the parent *bone* area. In other cases we leave the *joint* area as a separate segment.

### 5.5.6 Parameters

The level of detail can be controlled by choosing the $t$ and $\kappa$ values. Especially setting the $t$ for the basic skeleton is important as new edges can only be inserted into a joint that is marked as an *improvement area*. Those areas are detected if there are maxima inside. For bigger $t$, the ADF tends to have less maxima than for smaller values of $t$. $\kappa$ controls whether features that are not very thin, like humps, can be transformed into bones. Such a feature would usually produce an edge of the Reeb graph with a small difference between minimal and maximal function values, compared to thin features.

The other measure of level of detail - the feature length - is controlled by the $t$ for the smallest added skeleton.

The described parameters are not shape specific but rather general. For a defined set of parameters, all of the shapes can be processed by the skeletonization algorithm. Then the results can be the base for other geometric tasks - for example as a good similarity measure between the shapes.

## 5.6 Discussion and Future Work

We have proposed the ADF as an effective tool for shape analysis. It is governed by only one parameter, feature scale, which is arguably indispensable, and its construction requires nearly nothing more than a framework for computing eigenfunctions of the Laplace Beltrami Operator. Thus, it is also simple. We have also explored the application of the ADF to the related tasks of skeletonization which is shown in Figure 5.7 and segmentation which is shown in Figure 5.6. The result is an algorithm which has a controllable level of detail, insensitivity to noise and invariance to scale, rigid transformation and pose invariance. The last properties being inherited from the ADF.

In the future, we would like to explore other applications of the ADF. For instance matching features extracted as ADF maxima between shapes, and parameterization of shapes.

## 5.7 Acknowledgments

The 3D models used are provided courtesy of the Princeton and Aim@Shape repositories. We are very grateful to Steen Markvorsen for pointing out the connection to the Minakshisundaram-Pleijel expansion, to Julien Tierny for allowing us to use two of his images, and to the anonymous reviewers for insightful comments.

Chapter 6

# Tracing Reeb Graphs of the Auto Diffusion Function for Retrieval of Salient Shape Regions at their Best Scale

Katarzyna Wełnicka
Jakob Andreas Bærentzen

Technical University of Denmark

Presented as a poster on *Symposium on Geometry Processing Lausanne 2011.*

Reeb Graphs have proven to be a useful tool for analyzing shapes, especially when using functions related to meaningful geometric properties. Although some of the topological shape characteristics are function invariant, the function might be seen as a lens through which we look at shapes and the Auto Diffusion function is a lens with continuously changing focal properties. Depending on the time parameter, features of different scales can be spotted. This work aims to make use of this continuous multi-spectral property. We trace the evolution of Reeb Graphs obtained through the Auto Diffusion Function at a given time, while continuously changing the time parameter. Information collected through such tracing can be useful for many shape analysis tasks. We show an application where salient shape regions of different scales are retrieved.

## 6.1 Introduction

A lot of recent work in the shape processing community aimed at finding shape descriptors which are able to reveal the shape's general structure in an intuitive way. The intuitiveness can help in the computer user interaction when performing shape retrieval tasks such as searching for shapes having similar subparts or analyzing the structure diversity of the set of shapes within a given class.

Understanding the shape structure can be supported by studying scalar functions defined over the shape [17]. It can be seen as a way to capture the main shape features, for example through a Reeb Graph, which encodes the connected levelsets of the function into a graph structure. One function, however, might not be enough to capture all information about the shape. It is good to be able to combine information from many functions like in [20], where information from two Laplace Beltrami eigenfunctions is coupled together. In this article we present a method to integrate the information from a continuously parametrized family of functions. We achieve that through observing the evolution of a Reeb Graph while continuously changing the function.

The study of diffusion process on a shape resulted in many sound methods for extraction of meaningful pose invariant geometric features. Analyzing shapes by looking at heat diffusion from a point to itself was previously done either by fixing the points and changing time values which is known as Heat Kernel Signatures [143], or by considering a specific time value and looking at function

on the mesh known as the Auto Diffusion Function [54]. Our method combines both of these approaches by tracing the evolution of Reeb Graphs when changing the time parameter of the ADF.

We also introduce a simple algorithm for establishing correspondences between Reeb Graph edges through voting.

Matching edges of Reeb Graphs across the timescale leads to an application where the salient shape regions can be discovered together with the scale they belong to.

## 6.2 Related Work

### 6.2.1 Salient Features

The saliency approach usually appears in the context of the efficient shape retrieval or shape matching. It might be related to different geometric features such as: local shape descriptors evaluated at given points [55, 132], shape views [83], lines [23], or shape parts [50]. However, the general idea is similar: to chose a small subset of features, such that the complexity of solving the given problem can be decreased while keeping the performance at a similar level. The whole idea stems from the fact that many features do not contain any new information, so sparser sets of features can be used.

There are two approaches of evaluating the saliency with respect to a given feature. One of them is to measure their performance on the task they were intended to work for. The saliency here is identified with the best retrieval power performed on some training database [49, 83, 132].

The other approach is to connect feature saliency with geometric properties of the shape. A lot of work is based curvature related measures. [84] define mesh saliency in a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures. In [55] salient points are selected according to the uniqueness of their descriptor values; the descriptors are the volumes of the intersection of a ball at given points with the shape, calculated at different ball radii. Work of [50] performs an extraction of the salient geometric regions. Basic regions are constructed through fitting a quadric. Then they are merged as long as the saliency measure is increasing. This measure is combined from the curvature and area of the regions together with the variance of the curvature and number of the curvature local extrema. Authors aim to match shape parts,

which are numerically or topologically dissimilar, but approximate similar regions. Our approach might be seen as orthogonal to theirs (see fig. 6.1). We are more interested in general shape more than the local surface details. Many methods which rely on the distinctiveness and variability of curvature measures fail when a surface does not vary too much because of the lack of characteristic feature points. In some shape parts, like tubular regions, there might be no points of distinctive curvature characteristics, but those points together form an interesting part.

[40, 142] take the maxima of the Heat Kernel Signatures taken at a specific timescale. The most persistent are chosen, using the persistent homology method, when the filtration is given by the ADF function at a given scale. [142] picks the maxima points at high timescale and chooses the most salient ones as the base feature points for finding correspondences. [40] also uses the idea of the of the heat kernel signatures, but a small timescale is used, and then at the most persistent points, the heat kernel signatures are evaluated to form a feature vector used further for shape retrieval. Although persistence diagrams are stable [35], the location of the persistence points themselves might change drastically due to a 'winner takes all' scheme: if two maxima are to be merged the smaller one always dies even if it is just slightly smaller than the winner. A more accurate approach with provably stable results was presented by [137] in the field of segmentations. Instead of cancellation merging of the maxima influence areas occurs. Here, again, we have a situation where the calculations are performed at a given timescale. Our approach is similar in this manner that rather to taking specific maxima points of a function, parts of a surface, which belongs to the same topological areas with respect to heat kernel related function, are used. In our case the Reeb Graph Edges are taken, which allows us to extend analysis to more than sphere-like topologies.

### 6.2.2 Skeletons and Segmentations

Our work is related to skeletonization and segmentation but we do not perform a segmentation which partitions the into disjoint regions, as our regions can overlap. Also, it cannot be considered as a hierarchical segmentation - overlapping is not limited to inclusion, but features of one scale can cross the borders of features of other. We think that segmentation is not required for the partial shape retrieval applications and it is too constrained as it does not allow to have overlapping regions.

The work on segmentations of Golovinsky et al [59] and later by Chen et al [33] suggest that very often there is more than one good solution to the segmentation problem both when considering the placement of the cuts between different parts

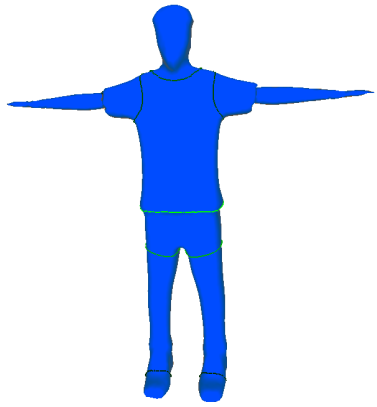


Figure 6.1: Starting level sets of the most edges with the highest total length are shown. The color indicates the best scale, which ranges from coarse (green) to fine (black). Note that the division in our method is not always done along the highest curvature but rather due to the general structure of the mesh.

the decision if some parts should be treated as separate.

Some multilevel skeletonization methods aim to extract features at many scales. The skeletonization method related to the medial axis transform was done by [56]. Anther approach [54] was to create the skeleton by integrating together Reeb Graph of ADFs computed at several scales. It starts from the level of the coarse timescale and inserts new edges into the skeleton if the mesh occupied by them was not taken by any edge, and if the edge length was big enough.

Like in the case of segmentations, one skeleton might not be enough to describe the shape when it contains multilevel information. A sticking out knee or a hump of armadillo or camel can be seen as an example of ambiguity situation that sometimes happens. The decision whether it should have its own representation in the skeleton or not is usually made by a skeletonization algorithm, and very often depends on the protrusion's size or its position within other shape parts. So, in one case the hump is created, and in another case, which has just a slightly smaller hump, it is not. Also details which are usually not included in the skeleton might be of interest for many shape retrieval methods.

## 6.3 Background

In this section we briefly describe the main concepts of the methods we build on: Reeb Graphs and the diffusion process on a manifold. More details can be found in the related literature.

### 6.3.1 Diffusion Kernel

Given a manifold $\Omega$, the diffusion kernel $K(x, y, t) : \Omega \times \Omega \times \mathbf{R}^+ \to \mathbf{R}^+$, or heat kernel, is a fundamental solution to the heat equation:

$$(\Delta_x + \partial_t)u(x; t) = 0$$

It can be evaluated [124, page 32] in the terms of Laplace Beltrami Operator eigensolutions as:

$$K(x, y, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t}\phi_i(x)\phi_i(y)$$

The kernel indicates how much heat from point $y$ is seen at point $x$ after time $t$ of the diffusion process. [143] and [54] propose to take $y = x$ which leads to a function defined on the $\Omega \times \mathbf{R}^+$ domain and describes how much heat from point $x$ remains at the same point after time $t$. [143, equation 4] fixed the point $x$ and defined the $\mathbf{R}^+ \to \mathbf{R}^+$ function to be the Heat Kernel Signature at a given point $x$ while [54, equation 1] fixed the time value $t$ and defined the $\Omega \to \mathbf{R}^+$ as the Auto Diffusion Function at a given time t, which was further analyzed through Reeb Graphs.

In order to be able to compare the diffusion values for shapes with different sizes, and to be able to compare the values across different times, normalization is necessary. We divide the function values by the heat trace at a given point [143] which is equivalent to the integral of the value over the whole surface. To get scale invariance, division of the time parameter by the first eigenvalue is used [54]. So, for Reeb Graph computations, we use the normalized values of the Auto Diffusion Function:

$$\hat{ADF}_t(x) = \frac{ADF_t(x)}{\int_\Omega ADF_t(x)} = \frac{K(x, x, \frac{t}{\lambda_1})}{\sum_{i=0}^{\infty} e^{-t\frac{\lambda_i}{\lambda_1}}}$$

### 6.3.2 Reeb Graphs

Given a manifold $\Omega$ and a Morse function $f : \Omega \to \mathbf{R}$, the Reeb Graph [19] is a quotient space of $\Omega \times \mathbf{R}$ defined by the equivalence relation $(p, f(p)) \sim (q, f(q)) \Leftrightarrow f(p) = f(q)$ and $p$ and $q$ are in the same connected component of $f^{-1}(f(p))$. Reeb Graphs trace the evolution of the function level sets defined on a manifold. Nodes of the Reeb Graph correspond to the critical points of the functions, and edges represent connections between them.

Many times we refer to the Reeb edge length and what we mean by this term is the difference of the function values evaluated at the endpoints of the edge.

## 6.4 Matching Sequence Continuously Parametrized Reeb Graphs

### 6.4.1 Reeb Graph Matching

It is important to remember that a Reeb Graph is more than a simple skeleton, as all points of the surface can be identified with a Reeb Graph structure. This means that given two Reeb Graphs of the same shape, we can identify two edges of a Reeb Graph through a vertex. This kind of identification casts a vote on the correspondence of those edges. If we have more than $\phi > 50\%$ of votes from the first edge to the second and vice versa, we establish a correspondence between them. Threshold $\phi$ being more than 50% guarantees that we have bijection between edges with an established correspondence.

The voting approach is suitable if the function that generates the compared Reeb Graphs does not change too much. We can assume that this happens when we sample the time parameter densely enough. To assume further correctness, which might be violated by plateau regions [137], edges with too small length are automatically set as unmatched.

We do not consider false negatives to be a problem. If there are edges which in reality are corresponding and remain unmatched by our voting process, it means that they change too much with the parameter. This usually happens with edges of a small edge length. The sensitivity of the edge tracer can be controlled by specifying the density of sampling the time and by the $\phi$ parameter.

False positives - edges which cover the same part of the shape but are not corre-

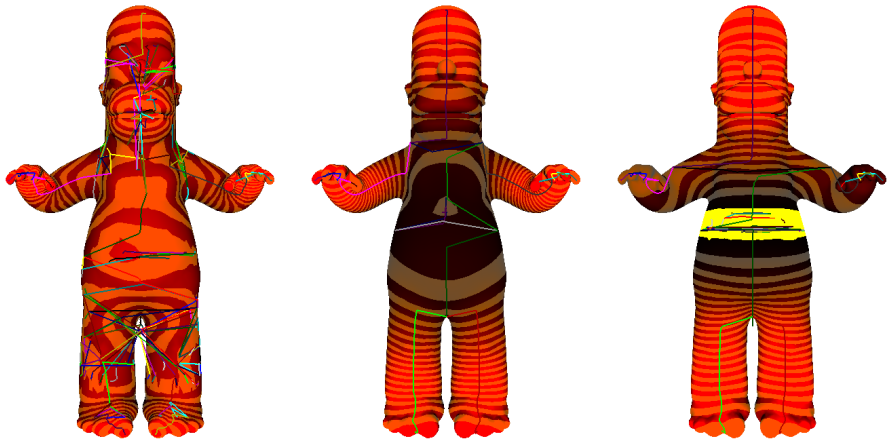


Figure 6.2: $A\hat{D}F_t = 0.05, 0.5, 5$ and Reeb Graphs with positions of the nodes placed at the centers of the corresponding level sets. Stripes indicate the level sets of the function and their density shows how much the function changes on the mesh.

sponding topologically - may happen if we do not sample the time scale with a proper density. In order to deal with such situations we check the graph's consistency after each matching (see also [62]). First, we remove from the skeletons all edges which were not matched in the voting stage. While removing edges, we merge their endpoints so the connectivity of the graph is kept. The result is two skeletons with a correspondence between their edges. Then a test is performed which checks if adjacent edges from one graph are mapped to adjacent edges of the other graph (edges to both endpoints incoming and outgoing from both endpoints). If the result of the test is negative, we construct an intermediate Reeb Graph, and perform recursive attempts to find the mappings until all intermediate graphs are consistent. Then the final matching is done by composition of the intermediate maps.

### 6.4.2 Evolution Track

We sample the time parameter in a logarithmic scale and establish the correspondence between the incident Reeb Graphs obtained by $A\hat{D}F_t$ functions computed at the sequence of time values. We identify the edges which were matched across the timescales and we call such identified set of edges as the

*diffusion-evolving Reeb edges.*

For our application we collect information about the $A\hat{D}F_t$ values at the endpoints of the edge at each time step. If a greater level of distinction is needed, there is also possibility to include other types of statistics [110] about the part of a surface identified with the edge. Also, some additional information can be provided - for example whether the edge has a sink or a source at one of its endpoints.

## 6.5 Discovering the Best Scale

When looking at the plots of the normalized Auto Diffusion Functions and their Reeb Graphs, we can see that at some time values the features of one scale are more distinctive while on other scale they disappear. When tracking the Reeb Graph through time, one can identify the time value at which the length of the edge reaches its maximum.

**Definition 6.1** *best scale* of a diffusion-evolving Reeb edge is the time at which it reaches a maximum length.

**Definition 6.2** *total edge length* of an edge is the length of a diffusion-evolving Reeb edge at its best scale.

**Definition 6.3** *best scale region* of the mesh is the surface corresponding to the diffusion-evolving Reeb edge at its best scale.

Usually, one can detect a huge number of best scale regions for a given shape, which is a consequence of many evolving edges (see figure 6.2). Some of them are not very informative when it comes to the description of the shape. With respect to the diffusion-evolving Reeb edges there are two types of importance measures of the edge which can also be seen as the edge 'persistence' - one is the lifetime of a given edge and the second one is the total edge length.

**Definition 6.4** The *$(\theta, \sigma)$-salient shape regions* of the shape are the best scale regions of the diffusion-evolving edges of the shape with lifespan between $t_b$ and $t_d$ such that $\frac{t_d}{t_b} > \theta$ and total edge lengths $> \sigma$

## 6.6 Experiments

For each shape we are to analyze we perform a tracking. The initial time is $t = 0.01$ and we take the new time value by multiplying the recent by 1.1 until we have $t = 10$, which results in 73 time samples. For all of them we calculate $\hat{ADF}_t$ like in [54] and extract the Reeb Graphs. The matching procedure is performed between consecutive time steps with $\phi = 0.56$ and for all matched scales we compute the best scale, total edge length and the best scale region. Then we can filter out the edges by using thresholds $\theta = 1/2$ and $\sigma = 0.01$

Figure 6.3 shows 20 most best scale regions corresponding to the diffusion evolving Reeb edges which have the highest total length. From these examples we can see that the total edge length can intuitively be identified as the protrusion level of a region while the best scale as the feature size with respect to the shape size.

We have also plotted (fig. 6.4) the values of: best scale and total edge length for 5 different classes of shapes. From the plot we can see that some of the points formed clusters. Moreover we can see that some of the parts share similarities across the classes - for example teddies and armadillos in area (2,40), women and armadillos (4,25) or dogs and horses (5,20). We can also see from this plot that there is a tendency that the diffusion-evolving edges with the highest total edge lengths have also a long lifespan.

Finally to demonstrate the usability for partial shape retrieval, in our last experiment we have calculated the descriptors for around 50 shapes. Then we collected all of the subparts in the database and queried according to the proximity in total edge lenght vs best area space.

The best timescale (BS) was coded as an index of a sample (ranging from 0 to 73). To combine it with total edge length (TEL) we computed the distance as follows:

$$d(i,j) = \sqrt{(TEL_i - TEL_j)^2 + \alpha(BS_i - BS_j)^2}$$

We set $\alpha = 64$. Some examples of salient region queries are shown in figures 6.5, 6.6 and 6.7.

## 6.7 Conclusions and Future Work

In this article we have presented how to match the Reeb Graphs of continuously parametrized functions. We introduced a method which is able to extract the

salient shape regions at their best scale by tracking evolution of the Reeb Graph while changing the time parameter for the Auto Diffusion Function. We have shown that similar regions can be retrieved by searching for the evolving edges with similar values of total edge length and the best scale.

We think that the results are promising. Still, a lot of further work can be done to expand the method. The matching procedure relies on common sense heuristics, it would be good to have more theoretical guarantees about its correctness. The births and deaths of diffusion-evolving Reeb edges resembles the barcode descriptor proposed by Carlsson et al [27]. They can also be plotted as a persistence diagram with birth and death times as coordinates, however diagrams are far from being stable. We believe that finding a topological structure whose elements can be matched across continuously changing parameter with a stable persistent diagram would be a great achievement.

In this work, we only treated the edges as independent parts. It is also possible to add second order information: the parent-child relationships between the evolving edges, which can be inherited from the relationships taken at a given Reeb edge. Also more advanced relationships which were not possible for the Graphs can be considered, for example up whether two edges which correspond to different scales are located at the same places of the mesh. This kind of enhancement results in something much more general than a skeleton.

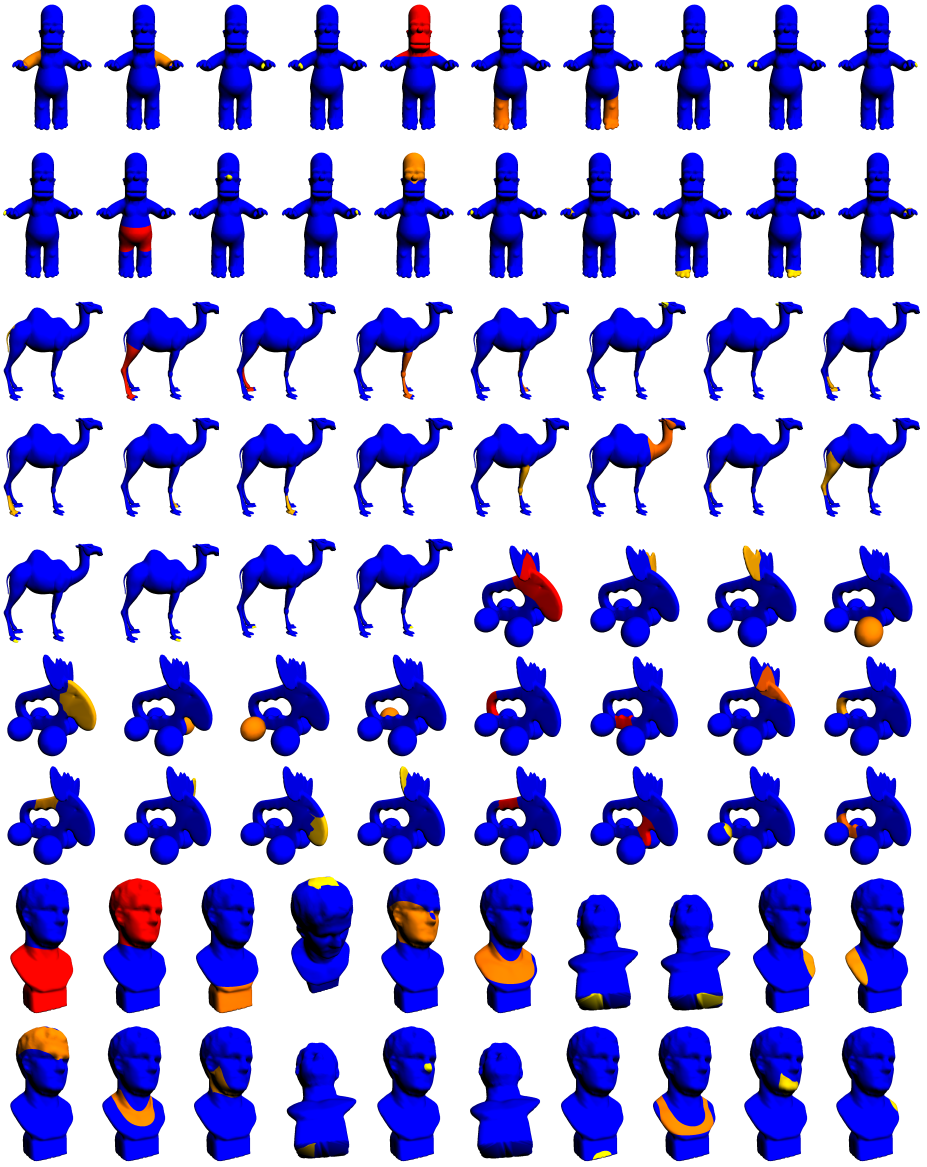


Figure 6.3: 20 best scale regions having the highest total edge lengths for homer, camel, elk and torso. The color indicates the scale of the space which changes from red ($t = 10$) to the yellow ($t = 10$).

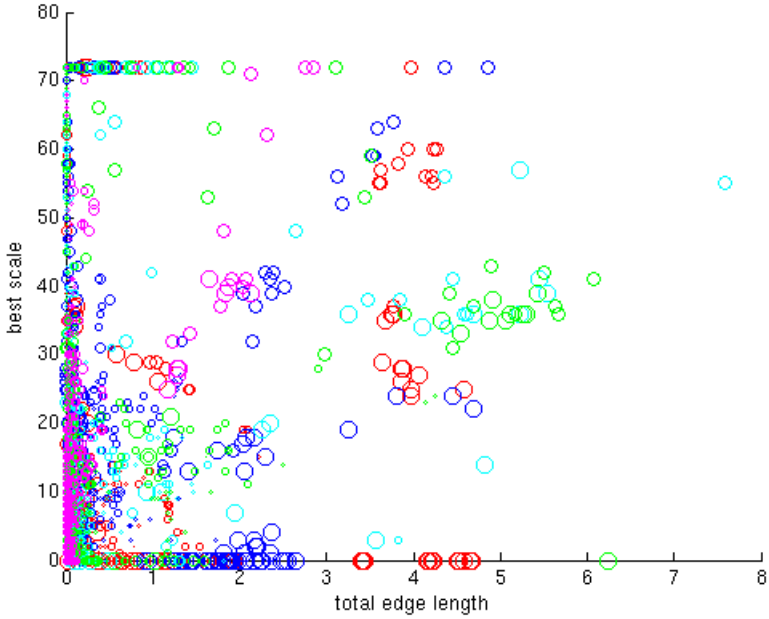



Figure 6.4: Best scale and edge length of evolving Reeb edges taken from the shapes of 6 women(red), 4 armadillos(blue), 4 dogs (cyan), 5 horses(green), 3 teddies(magenta). The clusters can be identified with corresponding body parts and area they cover with the variability of the proportions. The size of the bubble indicates the lifespan of an edge. Time sampling is done from $t = 0.01$ (0 on the scale axis) and then multiplied by 1.1 until it reaches 10 (72 on the scale axis).

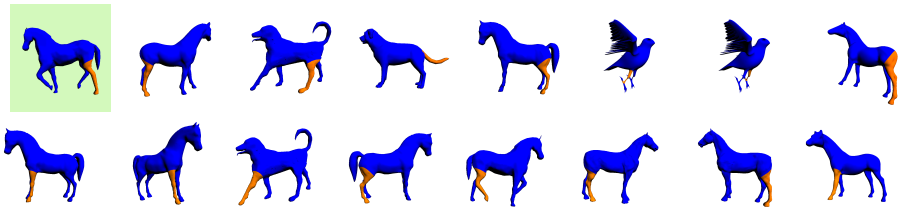


Figure 6.5: Retrieval of regions similar to horse's leg from a set of the salient regions of 67 shapes ('ant1' 'bird1' 'bird2' 'bird3' 'bird4' 'bird5' 'bird6' 'bull' 'camel' 'child1' 'child2' 'child3' 'cow1' 'cow2' 'cup1' 'cup2' 'cup3' 'cup4' 'dar1' 'dar2' 'dar3' 'dar4' 'dog1' 'dog2' 'dog3' 'dog4' 'doll' 'donkey' 'duck' 'elk' 'fish1' 'fish2' 'giraffe' 'homer' 'horse1' 'horse2' 'horse3' 'horse4' 'horse5' 'man1' 'man2' 'pig1' 'pig2' 'plane1' 'plane2' 'plane3' 'plane4' 'plane5' 'plane6' 'sci1' 'sci2' 'sci3' 'small' 'spider1' 'teddy1' 'teddy2' 'teddy3' 'teddy4' 'teddy5' 'teddy6' 'torso1' 'woman1' 'woman2' 'woman3' 'woman4' 'woman5' 'woman6')

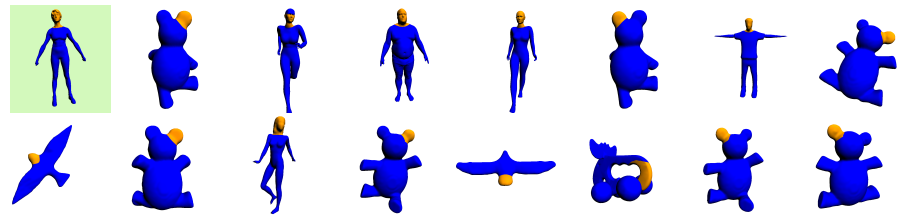


Figure 6.6: Retrieval of regions similar to woman's head like shapes from a set of the salient regions of 67 shapes

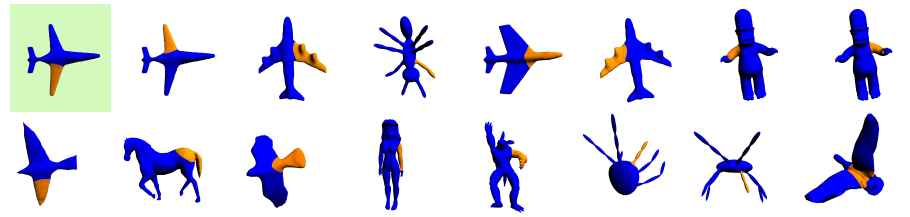


Figure 6.7: Retrieval of regions similar to plane's wing from a set of the salient regions of 67 shapes

Chapter 7

# Descriptor Based Classification of Shapes in Terms of Style and Function

Katarzyna Wełnicka
Jakob Andreas Bærentzen
Henrik Aanæs
Rasmus Larsen

Technical University of Denmark

*IMM-Technical Report-2011-17.* Contains extended version of:

[A] Katarzyna Wełnicka, Jakob Andreas Bærentzen, Henrik Aanæs, and Rasmus Larsen. Example based style classification. In Proceedings of the *Workshop on Mesh Processing in Medical Image Analysis (in conjunction with MICCAI 2011) Toronto, Canada September 18th, 2011.* Best paper award.

## 7.1 Introduction

It seems that great shifts in how human beings use technology often create a push for changes to the way we divide work between human beings and technology. Chemical film has all but disappeared and almost everybody takes digital photos which they proceed to put online for easy sharing with friends and family. Together with a number of other trends which have contributed to the vast amount of online and locally stored digital photos, this has made automatic recognition of people in images an important research topic - in spite of the fact that recognition is one of the tasks generally left to human beings, since we excel at recognition. We believe that recognition of the style of a 3D object is something that is also likely to be increasingly useful in the foreseeable future. Optical scanning methodologies make the generation of 3D content more feasible than previously, and it is easy to envision digital artists wanting to compile content for a 3D scene or composite object being in need of a method for searching for an object not just of a specific function but also a specific style.

The scope broadens further if we look beyond man made objects. It seems clear that, say, the various limbs of a specific human being have some commonality that separate them from those of another person. Thus, one could argue that an individual represents a style. Style in the context of biological variation is something that we explore in the work presented here. Specifically, we investigate whether we can define a style class for the teeth of a person.

Unfortunately, style is subtle and we do not hope to be able to automatically extract a description of style from 3D objects. Furthermore, we avoid using explicit ways of describing style. Recognizing the style of an object based on some textual or otherwise encoded information might be a feasible approach in some cases such as, for instance, recognizing to which order a given classical greek column belongs. But, relying on explicit information about a given style would require us either to solve the above problem of automatically extracting style information from shapes or to rely on human beings to encode style - a task that we believe would be both tedious and difficult.

Instead, we rely on examples in the work presented here. This requires that we have example (training) objects for each style. It also requires that we have an orthogonal class of functions, since, as we discuss below, the function of the object (what it is) clearly also has a profound impact on shape. Thus, our work can be summed up as example based classification of digital 3D shapes in both style and function categories.

### 7.1.1 Understanding style

Human beings excel at the task of recognizing objects, and, in fact, we are also very good at detecting the style of an object. However, an operational description of style that would allow a computer to detect the style of an object seems to be elusive, at least in general. On the other hand, it is encouraging that humans seem to be able to intuitively recognize the style of an object, and this is why we believe that it is feasible to attack the problem using statistical methods. Our hypothesis is that by using statistics of shape descriptors, we can compute properties of shapes and then train a classifier to discriminate between styles based on these descriptors.

But, there are two significant obstacles that need to be recognized and dealt with before proceeding.

1. Style is not a purely local or global effect. If only we could say to which parts of an object's geometry style has an influence, designing descriptors would be much easier. Unfortunately, that does not seem to be possible. If we think of style as being akin to a gene, the effect on the object's phenotype (metaphorically speaking) could be to the proportions of its parts, whether its edges and corners are sharp or rounded, whether smooth parts are curved or flat, whether it is embellished or not, or whether the surface is smooth or rough. Thus, we cannot simply use local shape descriptors and assume that these capture all the style information.

2. Perhaps a bigger problem is the fact that style is by no means the only thing that determines the shape of an object. Conversely, the function of the object would normally be the biggest contributing factor. By function we understand what the object is recognized as, i.e. the noun one would generally associate with the object, e.g. car, table, chair, tooth. Using a signal processing metaphor, the style is a, sometimes faint, signal superimposed on the stronger function signal.

From the first obstacle, we conclude that we need to use a broad range of descriptors in most cases and that we need to use descriptors which describe both local and global properties of shapes. From the second obstacle, we conclude that we cannot hope, in general, to achieve style discrimination if we do not take function into account. On a very abstract level, our approach is to first gain the ability to compute distances between pairs of objects in "descriptor space". This is a more general approach than always requiring that we have shape descriptors as, say, a multidimensional vector because we can sometimes compute a meaningful distance between two shapes directly (e.g. warping one to

| | $s_0f_0$ | $s_1f_0$ | $s_0f_1$ | $s_1f_1$ | $s_0f_2$ | $s_1f_2$ |
|---|---|---|---|---|---|---|
| $s_0f_0$ | 0 | 0.3 | 2.2 | 2.4 | 3.5 | 3.7 |
| $s_1f_0$ | 0.3 | 0 | 2.3 | 2.2 | 3.8 | 3.6 |
| $s_0f_1$ | 2.2 | 2.3 | 0 | 0.2 | 4.0 | 4.4 |
| $s_1f_1$ | 2.4 | 2.2 | 0.2 | 0 | 4.3 | 3.8 |
| $s_0f_2$ | 3.5 | 3.8 | 4.0 | 4.3 | 0 | 0.4 |
| $s_1f_2$ | 3.7 | 3.6 | 4.4 | 3.8 | 0.4 | 0 |

Figure 7.1: A matrix of exemplar distances between the shapes. Objects with the same function are much closer than the ones sharing the same style

the other) but not easily fix a set of coordinates for the two shapes in descriptor space. Given a set of objects where some objects share style and others function, a matrix of such distances might look as illustrated in Figure 7.1.

In this example style is indeed a fainter signal than function, and if we naively assume that the object with minimal distance to the knobby circle is another knobby object we are disappointed. As we see in Figure 7.2 it is the plain circle that is closest and then the knobby cube.

| | | | | |
|---|---|---|---|---|
| style | $s_0$ | $s_1$ | $s_0$ | $s_1$ |
| distance | 0.2 | 2.2 | 2.4 | 3.8 |

Figure 7.2: Style classification with a naive approach if we forget totally about the function and take into account only style. The shape would be classified to $s_0$, as those shapes seem to be closer than shapes having other styles.

Of course, this example is completely contrived and the distances made up. However, it corresponds well with our experience as our results will show, and if we do take function into account, we fare much better. Say, we have a query object of a given function and say we exclude objects of the *same* function, then for a group of objects that all share a common function (different from the function of the query object) the object(s) in that group which has the same style as the query object are likely to be closer to the query object than the objects which have a different style – in addition to having a different function. At least this is our hypothesis which is illustrated in Figure 7.3.

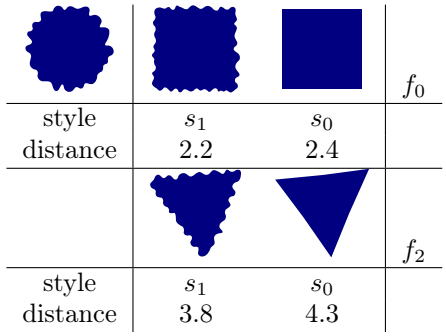



Figure 7.3: Because we grouped shapes to have the same function the difference in the distances is mostly caused by the style effect so the query shape is classified correctly as having the style $s_1$.

In summary, our hypothetical model is that

$$\text{Shape} = \text{Function} + \text{Style} + \text{Noise} \tag{7.1}$$

where addition should be construed as a more abstract composition operator. In the formula above, we readily admit that function probably has the greater influence. Taking that into account, we believe that we can approach the daunting task of creating a statistical model for style.

### 7.1.2 Related Work

Style and function separation in the context of man made three dimensional shapes was recently mentioned by Xu et al. [159], where the style of an object is defined by the proportions (anisotropic scaling) of its parts. It seems to be a very intuitive and reasonable approach but this does not exhaust the subject.

In many shape processing articles, even if the problem of style is not addressed in an explicit way there are situations where the space of given shapes is broken down into two different independent classification systems. In the deformation transfer [141] different kinds of animals can take similar poses, in which case it is quite easy to localize them, as the type of animal is described by an intrinsic metric of the shape surface, and the pose is its embedding in three dimensional space. The idea of geometric texture [4] fits within this framework as it aims to separate overall shape from its geometric details. Application of example based priors for surface reconstruction [52, 116] can also be seen as imposing style to the object.

In the image processing field, Hertzmann et al. [60] presented a method that, when given three images: one with style $A$ and function 1, one with style $B$ and function 1, and another one with style $A$ and function 2, creates an image with style $B$ and function 2. The same concept has also been explored by this group in the field of curve styles [61]. Other related problems can be found when dealing with images of fonts, separating lighting conditions from the scene and distinguishing between a spoken language and the accent. All those three cases were examined through bilinear models by Tenenbaum et al. [147].

Tenenbaum's framework requires establishing one to one correspondences both for the style and for the function - for example fonts are compared through corresponding pixels of their bitmaps. In general, for different types of shapes obtaining such correspondences can be difficult. Similar corespondances need to be established across the styles for Hertzman's work. Our approach does not require any correspondence finding, which is usually costly and sometimes outright impossible, like in the problem of registering a table to a chair [57].

In general, if the feature space is available, many well established statistical methods can be used for metric learning. For example Linear Discriminant Analysis [96], which modifies the feature space such that, for a given training set containing objects from different classes, the inter class variance is maximized and intra class variance minimized. A similar approach was also used by [157] which makes it possible to define similarity and dissimilarity relationships between selected pairs of objects. In a similar manner, Giorgi et al. [57] customize a way of combining a set of distances between shapes so that user defined similarity is captured. In this work the metric is modified in order to reflect the user defined constraints of nearby or far away shapes. The final metric is taken as a maximum distance from distances given by all of the metrics, however, the particular metrics are scaled according to a similarity feedback provided by the user.

For the task of finding the replacement of an object from the one of the most similar style, a similarity measure needs to be established. As an input to the algorithm we have many hypothetic distance measures, but we do not know which one is the most suitable - having such information is indeed equivalent to solving the initial problem. We asses the relevance of a specific shape descriptor indirectly as a consistency requirement: dissimilarity or similarity between the styles should be reflected in a similar way for different functions. A similar methodology, based on indirect consistency, can be found in [162]. The aim is to remove incorrect mappings between different views of a scene. The quality of these mappings is assessed by analyzing the mapping loops which they form. An inconsistent loop indicates that at least one of the mappings that belong to it is wrong, while a consistent loop means that all mappings are likely correct. Having evaluated the correctness of many loops the bad mappings are spotted

through loopy belief propagation. A similar approach is performed by [106] in the space of shape maps.

### 7.1.3 Contributions

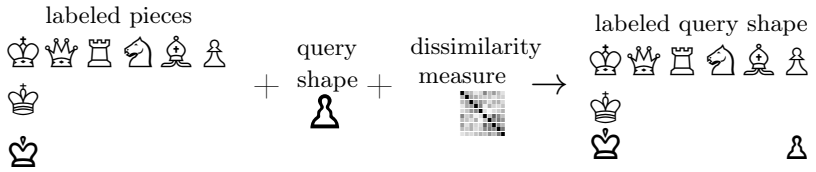
We introduce a framework for dealing with style by validating how given descriptors relate to style within the context of a given dataset (section 7.3). As far as we know there was no general framework for such problem, especially when no parameter space is available. We also show how we can deal with style classification, even if we cannot find the descriptors which are purely responsible for the style, by using additional function information (section 7.2).

We propose a statistical model for style which takes both function and style into account. Using this model, we can sort objects according to both style and function provided we have example objects for each style and each function. Operationally, this allows us to achieve some tasks. For instance, automatic sorting of chess pieces according to the type of chess and according to the set of pieces to which it belongs. We also solve the task of finding a tooth model which can be used for the design of a prosthesis to replace a missing tooth – even if we do not have a very similar tooth model in the patient's mouth (section 7.5.3).

The general methods are introduced in sections 7.2 and 7.3. In section 7.4 we present the descriptors used for our experiments and in section 7.5 we show the experiments performed on tooth and chess datasets and analyze the obtained results.

## 7.2 Style and Function Classification

In this section we will show how information about style and function hidden in the same metric can be decoupled. The problem is illustrated in figure 7.4. We assume here that we have a dissimilarity measure that was produced by some kind of shape descriptor and contains both functional and stylistic information. One example of such a measure was given in the introduction section (see figure 7.1). Based on this information we want to be able to detect the most likely style and the most likely function of a query shape. Because we do not have explicit descriptions of the style or of the function, we assume they are given by the examples through the training set $T$.




**input** labeled pieces with at least one set of all functions and one function of all styles, shape of unknown function and style, distance measure

**task** find the most likely style and function of a query shape.

Figure 7.4: Diagram shows the task of classifying a shape according to style and function. Note that in this framework we need example shapes which serve as definitions of styles and functions and a way to measure the distances between the shapes.

### 7.2.1 Likelihoods Computation

In this section we introduce the likelihood of some shape being of given style and function, which is based on the distances of the unknown shape to shapes from a training dataset. We want to formulate it in a way that the function effect is eliminated when assessing the style and the style effect is eliminated when assessing the function.

We denote the shape of style $s$ and function $f$ as $S_f^s$ and the set of shapes of a function $f$ and style different than $s$ as $S_f^{-\mathbf{s}}$.

Our main observation here is that if we have in the training set shapes which share the same function but have different style, then it is possible that we may factor the function out as it was shown in figure 7.3.

For example if the distance to a $S_a^1$ is smaller than a distance to a $S_a^2$, we may say that the unknown shape is more likely to be of a style 1 than to be something else.

The partial likelihood $l_i^{s=j}(x:k)$ of unknown shape x to be of style $j$, when we have two example shapes of the same function $i$ of which one $S_i^j$ is of a style $j$ and another one $S_i^k$ is of a style $k$ other than $j$, is equal to:

| shape | $S_c^0$ | $S_c^1$ | $S_c^3$ | $S_c^3$ |
|---|---|---|---|---|
| j | 0 | 1 | 2 | 3 |
| $d(S_s^3, S_c^j)$ | 2.4 | 2.7 | 2.1 | 2.7 |
| $l_1^{s=j}(S_s^3)$ | -0.2 | -0.9 | 0.8 | -0.9 |

Figure 7.5: Likelihoods calculated within a training set of circles for a query object $S_s^3$. The biggest value is reached when the distance is the smallest

$$l_i^{s=j}(x:k) = d(x, S_i^k) - d(x, S_i^j).$$

In our training dataset $T$, for a given function $i$, we may have more than just one shape not being of a function $j$ so we take the mean plus the minimum of all of the partial likelihoods:

$$l_i^{s=j}(x) = \underset{k\in\{-\mathbf{j}\}:S_k^i\in T}{\text{mean}} l_i^{s=j}(x:k) + \underset{k\in\{-\mathbf{j}\}:S_k^i\in T}{\min} l_i^{s=j}(x:k).$$

Note that minimum is equal to the distance to the closest of the known shapes from function $i$ and style other than $j$, minus the distance to $S_j^i$. If the style $j$ is the closest of the shapes from that style, then the minimum will be positive, otherwise it will be negative. The mean value stabilizes the results by taking into account distance measures of all of the shapes of this function but different style. So the first term is an average distance of the unknown shape to a shape from the training dataset having a function $i$ and not being the style $j$ minus the distance to $S_j^i$.

The use of differences of the distances instead of the direct use of distances is caused by the additive model expressed in the equation 7.1. Because the distance of two shapes is caused both by functional and stylistic differences, we want to remove the impact of the function by using the difference of distances to shapes having the same function but different styles.

Such an approach is based on the assumption that the distance to a shape having different both style and function properties should be greater than the distance to a shape being different just with respect to style (or function). Graphically speaking, if we put two shapes of the same style in the same row and two shapes of the same function in the same column of a table, it means that the distances computed across the diagonals would always be greater than the ones between shapes displaced only vertically or horizontally (see fig. 7.6).

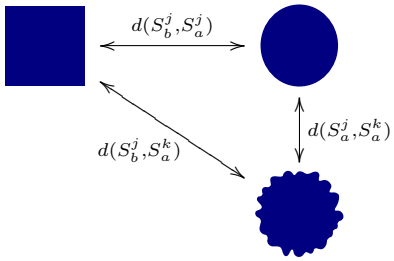



Figure 7.6: Graphical representation of the diagonal property. The distance on the diagonal should be greater than the vertical and horizontal distances. In this way even when the inter-style (vertical) distances are much smaller then the inter-function (horizontal) distances we are still able to capture the stylistic difference, when we know function labels.

**Definition 7.1** The metric $d$ has a *diagonal property* on the set of shapes $\mathbf{S} = \{S_f^s : f \in F, s \in S\}$, if for any $j, k \in S$ and $a, b \in F$ we have

$$d(S_a^j, S_b^k) \geq d(S_a^j, S_b^j)$$
$$d(S_a^j, S_b^k) \geq d(S_a^j, S_a^k)$$

**Lemma 7.2** *If $d$ on $\mathbf{S}$ has a diagonal property, then for $S_h^j \in \mathbf{S}$ for all $k \in -\mathbf{j}$ we have $l_i^{s=j}(S_h^j) \geq l_i^{s=k}(S_h^j)$*

Proof. Let's denote the mean part of the $l_i^{s=j}(x)$ as $l_i^{s=j}(x)_{\text{mean}}$ and the minimum part as the $l_i^{s=j}(x)_{\min}$. So we have $l_i^{s=j}(x) = l_i^{s=j}(x)_{\text{mean}} + l_i^{s=j}(x)_{\min}$.

For the mean part:

$$\begin{aligned} l_i^{s=j}(x)_{\text{mean}} &= \frac{\sum_{m \in -\mathbf{j}} d(x, S_i^m)}{|i|_T - 1} - d(x, S_i^j) \\ &= \frac{\sum_{m \in -\mathbf{k}} d(x, S_i^m) + d(x, S_i^k) - d(x, S_i^j)}{|i|_T - 1} - d(x, S_i^j) \\ &= l_i^{s=k}(x)_{\text{mean}} + \frac{|i|_T}{|i|_T - 1}(d(x, S_i^k) - d(x, S_i^j)) \end{aligned}$$

Where $|i|_T$ denotes the number of shapes in the training dataset that have a function $i$. By putting $x = S_h^j$, and having $d(S_h^j, S_i^k) - d(S_h^j, S_i^j) \geq 0$ from the diagonal property, we get:

$$l_i^{s=j}(S_h^j)_{\text{mean}} \geq l_i^{s=k}(S_h^j)_{\text{mean}} \tag{7.2}$$

$$l_i^{s=j}(x)_{\min} = \min_{m \in -\mathbf{j}} d(x, S_i^m) - d(x, S_i^j)$$
$$= \min\left(d(x, S_i^k), \min_{m \in -\mathbf{j} \cap -\mathbf{k}} d(x, S_i^m)\right) - d(x, S_i^j)$$

analogically

$$l_i^{s=k}(x)_{\min} = \min\left(d(x, S_i^j), \min_{m \in -\mathbf{j} \cap -\mathbf{k}} d(x, S_i^m)\right) - d(x, S_i^k)$$

Applying a diagonal property $d(S_h^j, S_i^k) \geq d(S_h^j, S_i^j)$ and using the fact that $a \geq b$ implies $\min(a, c) \geq \min(b, c)$ for any $c$ we have:

$$l_i^{s=j}(S_h^j)_{\min} \geq l_i^{s=k}(S_h^j)_{\min} \tag{7.3}$$

And from inequalities 7.3 and 7.2 we have also: $l_i^{s=j}(S_h^j)_{\min} \geq l_i^{s=k}(S_h^j)_{\min}$

In order to gather information from all of the training functions, we take the mean value plus the maximum of all the styles for which in the training set there is a style $j$ and some shapes not being of style $j$.

$$l^{s=j}(x) = \max_{i: S_j^i, S_{-j}^i \in T} l_i^{s=j}(x) + \operatorname*{mean}_{i: S_j^i, S_{-j}^i \in T} l_i^{s=j}(x).$$

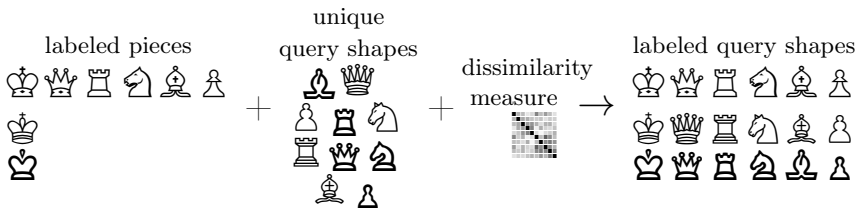
Here, by taking the maximum, we are favoring the function for which the style j is most likely. The mean is again added to get the distance information from all known styles.

There might be cases when we do not have enough information in the training set for establishing likelihoods. This happens when there is no set which has a training representative for the style $j$ and for some shape which is not of a function $-j$. In such a case we set the likelihood to zero.

The whole problem might be inverted and the likelihood computation of "x being the function i" is done in an analogous manner. Then for a given x the cost of assigning to it style j and function i is equal to: $l_{f=i}^{s=j}(x) = l_{f=i}(x) + l^{s=j}(x)$.

### 7.2.2 Exploiting Uniqueness

If the dissimilarity measure on the given dataset has the diagonal property then the likelihood will never fail to show the style and function of an unknown object. In many applications this is not the case: for at least some percentage of style and function pairs the property will be violated. So the correct style or the correct function will not get the largest likelihood score. If we however address




**input** labeled pieces with at least one set of all functions and one function of all styles, set of missing shapes of unknown function and style, distance measure

**task** find the best configuration of the query set, according to labeled set.

Figure 7.7: Diagram shows the task of sorting the query shapes. Each shape can be assigned only to one label. This constraint makes the task much easier than when treating all query shapes separately.

the problem not in separate queries but assume that we need to assess the style and function for a bunch of shapes (figure 7.7) of which we know that there are no repeating shapes in our query then our results can be much improved.

This approach follows a general concept that a uniqueness constraint makes the problem much easier. We use the likelihoods as negative costs and solve the minimum linear assignment problem [25] for the unknown labels and loose shapes.

#### 7.2.2.1 Multiple Step Assignment

An assignment problem with the costs defined above does not make use of the information about all of the distances between the shapes. Only information about the distances to the training shapes is taken into account. It might be an advantage when we do not want to compute the distances between all of the shapes, but if we already have computed all of the distances we may want to include them to make our algorithm better.

If we are able to locate the shapes for which we can expect that the initial matching went correctly we can add those into the training dataset with the labels obtained by the initial assignment. We do not have an oracle which tells which pieces were assigned correctly and which were not. If we had such

an oracle it would also automatically solve our assignment problem. However with some additional measures we can assume that there are pieces which were labeled more reliably than the other ones.

In order to estimate the labeling reliability, we calculate the diagonal cost of the assignment which we define as the average sum of similarities between all the shapes having the same style or function labels.

**Definition 7.3** Let $A(S)_i^j$ denotes the shape S which has a function label $i$ and style label $j$ obtained by an assignment $A$. Then a *diagonal cost* of such assignment is:

$$dc(A) = \frac{\sum_{A(S)_k^m} \left( \sum_{A(S)_k^{j \in -\mathbf{m}}} d(A(S)_k^m, A(S)_k^j) + \sum_{A(S)_{i \in -\mathbf{k}}^m} d(A(S)_k^m, A(S)_i^m) \right)}{\sum_{A(S)_k^m} \left( \sum_{A(S)_k^{j \in -\mathbf{m}}} 1 + \sum_{A(S)_{i \in -\mathbf{k}}^m} 1 \right)}$$

For a hypothetical unknown shape we might add it for a moment to the training set with the labels that we got as the solutions of the initial problem. Then as the labeling reliability we could calculate what is the diagonal cost when assignment is solved with the use of this piece. However, we discovered that instead of calculating diagonal costs directly it is better do the inverse assignment, which is performed by swapping the unknown data with the known, solving the inverse problem and then calculating the diagonal cost.

It might happen, for example if we know all the shapes of two styles, which become fully unknown in the inverse problem (fig. 7.8), that we are not able to find the inverse solution. In such case we take small subsets of data, by excluding from the problem all but one of the shapes of the style (function), which for the inverse problem have the style (function) cost undetermined. We solve inverse subproblems and take the sum of the diagonal costs for all of the given subtasks, divided by the number of all elements sharing the style and the function. The smaller the inverse diagonal cost, the more reliable is the hypothetical assignment of the unsorted shape to its label.

We also added other sanity checks of the shapes and consider the following properties:

**swapping minimum** : swap the hypothetical shape label with all other shape labels in an initial assignment. If the diagonal cost of some of the swapped assignments is smaller then the initial one, this piece is unreliably assigned.

**perturbation persistence** : solve two assignment problems as initial but in-

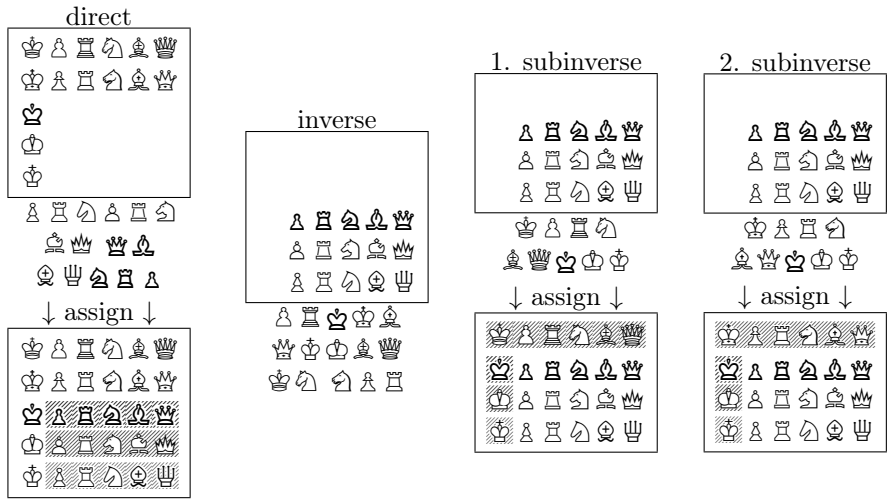



Figure 7.8: The inverse problem is made from a solution of a direct problem by exchanging the known information with the unknown. Because the solution of the inverse problem is not always unique, we instead solve the subproblems and then calculate their diagonal cost, which then is used to calculate the final inverse diagonal cost.

stead of original cost use

$$l^{s=j}_{f=10i}(x) = 10 * l_{f=i}(x) + l^{s=j}(x)$$
$$l^{s=10j}_{f=i}(x) = l_{f=i}(x) + 10 * l^{s=j}(x).$$

Then if the chess piece is not assigned to the same label as the initial problem assignment of this piece is unreliable.

In order to minimize bad choices we always take the piece having minimum inverse diagonal cost and which is reliable according to the above reliability criteria. We add it to the training dataset and repeat the assignment and addition of the most reliable pieces until there is no reliable piece to be added. Then we use the assignment from the last step as the final assignment.

## 7.3 Consistency Learning

In this chapter we want to solve a problem which can be seen as a reverse to the one solved in the previous section. We have an incomplete set of shapes of

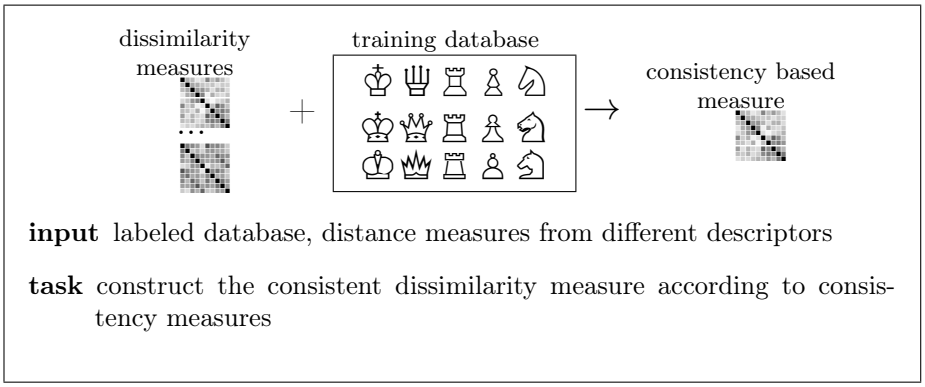



Figure 7.9: The training phase of the replacement finding task. The consistency dissimilarity measure is constructed based on consistency performance of different descriptors within the context of the training database.

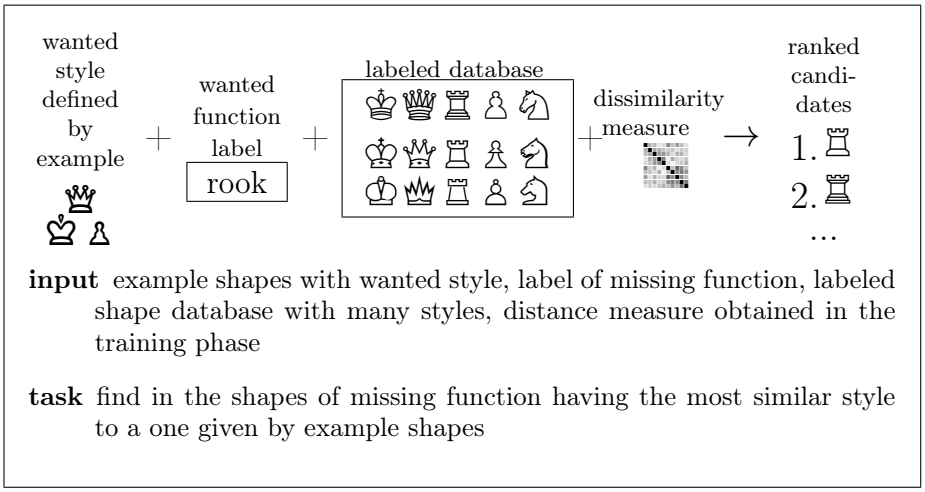



Figure 7.10: The second step of the replacement finding task. User comes with exapmles of a style and asks for a shape of that style and a function that was not present in the example shapes. The candidates are selected from the database and ranked according to estimated similarity to the queried style.

some style and we want to find the missing one. Our task is to search in the database of available shapes for the one which is most likely to be of a given style (figure 7.10).

We assume here that the function is known or can be easily detected. In many contexts function can be given explicitly: for example the type of a tooth is usually associated with its position in the mouth. Also, this is a sound approach when function is much more distinctive and we can determine it easily by using standard shape retrieval methods, as for example it is not so difficult to distinguish between a table and a chair.

We treat the 'style space' in a continuous way. We expect some styles to be very close to other styles, such that a shape of a given function can be replaced with a similar style quite well. This approach is especially suitable in a domain of shapes with some kind of biological variation.

In the previous section we took a good dissimilarity measure for granted. In general, we might not know it advance. Instead, we have many proposals of dissimilarity measures $d_i(,)$ which can be obtained through different kinds of shape descriptors $D_i$.

The task is to choose such a measure $d_i$ or a combination of measures with which we can distinguish between different styles. This task is related to metric learning approaches like LDA [96]. But the aim of LDA would be to have objects of the same style close and those of different style far away. We also require that the dissimilarity measures should be consistent across different functions.

We can illustrate the problem with the tooth shapes. Suppose a patient has one tooth destroyed. In order to be able to reproduce its shape, we want to find in a database a tooth which is mostly similar to the existing tooth he has. We have a molar missing but because a premolar is still in the patient's mouth, we wish to search in our database for a mouth which has the most similar premolar to the patient's. From that mouth we take a molar as a template for our new tooth. This approach assumes that similarity for premolars induces a similarity between molars.

The tooth replacement example shows that the consistency requirement is necessary as it aids in many concrete tasks – like searching for the best replacement for some missing data. Here we do not know directly what 'close' means, as we have many metrics but don't know which one is a correct. Usually a correct metric combination in such a case can be found by giving example pairs of shapes which are similar and pairs which are dissimilar [57]. In our case we do not have such information. Instead we can impose the metric consistency requirement: the distances between shapes having different styles and function $A$ should be close to the distances of the shapes of the same styles and function $B$.

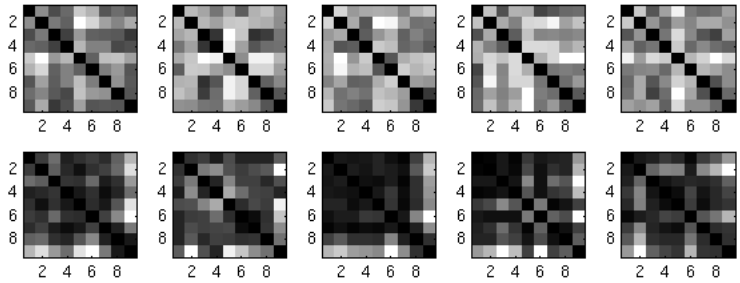



Figure 7.11: Distances between shapes of the same function and different styles. 5 functions are displayed (columns) and two different descriptors (rows). The top descriptor has consistency 19.35 and the bottom one has consistency 69.27. Note that the smaller the consistency measure, the more similar should the plots of different functions be. Data was taken from calculation for three dimensional descriptors of the chess pieces.

### 7.3.1 Consistency

In this section we introduce consistency factors which measure the similarities of distances between styles across different functions. We also create a final dissimilarity measure from different descriptors with the weights assigned according to the descriptors' consistency.

Assume we have a set of training shapes $S_i^j$, where $i = 1..n_i$ indicates the function and $j = 1..n_j$ the style. We also have $k_n$ potential dissimilarity measures $d_k(,)$

Let us take all distances $d_k(S^{i_1}_{j_a=1..n_j}, S^{i_1}_{j_b=1..n_j, j_b \neq j_a})$ between different shapes of the function $i_1$. In order to be comparable those distances need to be normalized, which we do by dividing them by the median of obtained distances. This results in a $\binom{n_j}{2}$ dimensional vector of k-distances between all possible pairs of shapes with different styles and the function $i_1$, which we will denote $v(d_k, f_{i_1})$.

For each pair $i_1 \neq i_2$ of two different functions we can establish the **consistency score** $cs^{i_1,i_2}_{d_k}$ with respect to a distance $k$ and function $i_1$ and $i_2$ as the norm of difference of distance vectors:

$$cs^{i_1,i_2}_{d_k} = \sqrt{\sum_{l=1..\binom{n_i}{2}} (v(d_k, f_{i_1})_l - v(d_k, f_{i_2})_l)^2} \tag{7.4}$$

In order to calculate the total consistency factor ($TCF_k$) for a dissimilarity

measure $k$, we take the sum of the differences for all function pairs. Note that the smaller $TCF_k$ is, the more consistent $d_k$ is with respect to style.

We construct the final measure by summing the dissimilarities obtained through different shape descriptors with weights that promote consistency.

$$D_f(x,y) = \sum_k e^{(-2\frac{TCF_k}{mean(TCF)})} \frac{d_k(x,y)}{\sigma_{d_k}} \tag{7.5}$$

where $\sigma_{d_k}$ is the median distance from distances $d_k(,)$ between all training shapes.

### 7.3.2 Query Based on Consistency Learning

In the training phase, given the database labeled with styles and functions, we compute the consistency measures from different descriptors and construct a final metric according to equation 7.5.

In the user phase, when asking for a specific function $q$ we also provide samples of the style with shapes labeled by functions: $Q_1, ..Q_n$. For each such shape, we measure the distances between $Q_i$ and all the shapes from the database sharing this function. The distances reflect the similarity of the queried style to known styles and they might be slightly different if the consistency of the final metric is not perfect. The measure of similarity of the queried style with styles in the database is obtained by summing $D_f$ for different functions.

$$D^a(s=j,Q) = \sum_i D_f(S_i^j, Q_i)$$

We take from the database the closest style as the one with the smallest $D(s = j, Q)$, and take the shape $S_q^j$ as a replacement of the unknown shape $Q_q$.

We can also go a step further and not take all $D_f$ with the same weights. Having the final dissimilarity measure we compute consistency scores $cs_{D_f}^{i_1,i_2}$ between different functions. These consistency measures can be used in order to asses what pairs of functions are better correlated. For example two neighbor upper molars can be more correlated than a molar and incisor. So if a molar is missing and we have the neighbor molar and incisor, we should give higher weight for query of the closest mouth with respect to a molar than with respect

to an incisor. We could either use only the distances with respect to a function most correlated with the query function, or use the weights according to the consistency scores:

$$D^w(s=j,Q)=\sum_i e^{\left(-2\frac{cs^{q,i}(D_f)}{mean_i(cs^{q,i}(D_f))}\right)}D_f(S_i^j,Q_i)$$

Both in the training and user phase we need a dataset, with labeled functions and styles. This can be the same dataset. The queried style needs to excluded from the training phase as we do not know what kind of data will be provided by the user.

## 7.4 Computing Distances Between Shapes

The methods presented in the previous sections are quite general and are independent of the descriptors we use. On the other hand the performance of the method relies on their choice, as if we use descriptors that are totally unable to capture similarities then the results we obtain will also be of poor quality. For example using only global descriptors for a dataset with the style being mostly expressed in local details would not be a good idea.

Note that in this paper we do not require for the dissimilarity measure produced by our descriptors to have properties of a metric space.

### 7.4.1 Curves Comparison with Dynamic Time Warping

In this section we explain how distances between oriented curves can be obtained through Dynamic Time Warping. This is a good example that there are cases when we have a way of establishing the dissimilarities between shapes without a general feature space. On the other hand there is always a way to go from a feature space with coordinates to a dissimilarity measure, by using one of many available vector norms. So the use of similarity is more general than the use of the feature space (see also Giorgi et al. [57]).

The curve $\mathbf{X}$ is represented as a polygonal chain $\mathbf{x_{i=0..n}}$. The standard dynamic time warping problem for two curves $\mathbf{A}$ and $\mathbf{B}$ is to find a sequence of correspondences between their vertices $\mathbf{a_{i=0..n}}, \mathbf{b_{i=0..m}}$, denoted as $\mathbf{C}=\mathbf{c_{i_k j_k}}$ where $\mathbf{i_k}\in\{\mathbf{0..n}\}$ and $\mathbf{j_k}\in\{\mathbf{0..m}\}$ and satisfying the conditions of:

**monotonicity** if $\mathbf{c_{i_k j_k}}, \mathbf{c_{i_l j_l}} \in \mathbf{C}$ and $\mathbf{i_k} \leq \mathbf{i_l}$ then $\mathbf{j_k} \leq \mathbf{j_l}$

**continuity** for incident correspondences $\mathbf{c_{i_k j_k}}$ and $\mathbf{c_{i_{k+1} j_{k+1}}}$ we have:
$\mathbf{i_{k+1}} - \mathbf{i_k} \leq \mathbf{1}$ and $\mathbf{j_{k+1}} - \mathbf{j_k} \leq \mathbf{1}$

Classical DTW searches for the correspondence with minimal sum of lengths of vectors $\mathbf{v_{ij}} = \mathbf{a_i} - \mathbf{b_j}$ whose endpoints are defined as vertices indicated by the correspondence.

$$\mathbf{d_{DTW}}(\mathbf{A}, \mathbf{B}) = \min_{\mathbf{C(A,B)}} \sum_{\mathbf{c_{i_k j_k}} \in \mathbf{C(A,B)}} \|\mathbf{v_{i_k j_k}}\|$$

The translation invariant version of Dynamic Time Warping, instead of minimizing the sum of lengths of $\mathbf{v_{ij}}$, minimizes the sum of lengths of the difference of vectors $\mathbf{v_{ij}}$ for two incident correspondences:

$$\mathbf{d_{TIW}}(\mathbf{A}, \mathbf{B}) = \min_{\mathbf{C(A,B)}} \sum_{\mathbf{c_{i_k j_k}} \in \mathbf{C(A,B)}, \mathbf{k>0}} \|\mathbf{v_{i_k j_k}} - \mathbf{v_{i_{k-1} j_{k-1}}}\|$$

The discrete version of DTW depends heavily on how the vertices are positioned on the curve, since for a given vertex the corresponding point must be chosen from the vertices of the second curve.

We use the method of Efrat et al. [45] and transform the translation invariant DTW into a continuous setting. In such a case we want to represent as $\mathbf{a_i}$ any point on a polygonal chain $\mathbf{A}$ and for this purpose we extended linearly the index $\mathbf{i} \in \{\mathbf{0}...\mathbf{n}\}$ to a domain of real numbers $\mathbf{0} \leq \mathbf{i} \leq \mathbf{n}$. This is done using the interpolation of values known at vertices: $\mathbf{a_i} = (\lambda - 1)\mathbf{a_{\lceil i \rceil}} + \lambda \mathbf{a_{\lfloor i \rfloor}}$ where $\lambda = \frac{\mathbf{i} - \lfloor \mathbf{i} \rfloor}{\lceil \mathbf{i} \rceil - \lfloor \mathbf{i} \rfloor}$. As a result vectors $\mathbf{v_{ij}} = \mathbf{a_i} - \mathbf{b_j}$ are also extended to a two dimensional surface defined as the combinatorial manifold $\mathbf{V}(\mathbf{A}, \mathbf{B}) = \mathbf{A} \oplus -\mathbf{B}$. After this modification the translation invariant problem can be defined as finding the shortest monotonous path $\mathbf{P}(\mathbf{A}, \mathbf{B})$ on this manifold which connects the endpoints $\mathbf{v_{00}}$ and $\mathbf{v_{nm}}$. The minimized function $\mathbf{d_{CTIW}}$ is the length of this path and this value is used to establish the dissimilarity between the curves.

### 7.4.2 Three Dimensional Descriptors

Recently there has been a lot of work within the content based shape retrieval field and a huge amount of different shape descriptors exist [145] and many new methods are proposed each year. The performance of such methods is

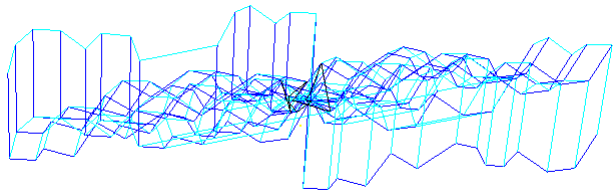


Figure 7.12: A manifold $V(A, B)$ when comparing two similar curves. Geodesic between the $\mathbf{v_{00}}$ and $\mathbf{v_{nm}}$ is marked with black. Although the embedding of this manifold is complicated it has a simple square topology

measured through the ability to retrieve objects of the same class as the query shape, across some given benchmark of shapes (SHREC retrieval contest). As mentioned by Godil et al. [58] in the field of general shape retrieval, usually hybrid methods, which combine many shape descriptors, perform better as they can capture many local and global characteristics.

Our main contribution was not to introduce any kind of a new descriptor which will perform well within our style-function problem. Instead we propose a method to asses the usability of existing descriptors, to combine and use them so that our style function discrimination tasks can be achieved.

Besides our general approach, we briefly present the descriptors which we have used as input for our methods. We have chosen to use local shape descriptors which rely on neighborhood at some distance from a given position. This way, by changing the neighborhood size both local and global features can be captured. We used three types of such descriptors:

**curvatures** : minimum and maximum curvature obtained by fitting primitives through points sampled from the neighborhood area [146] (2x4 descriptors),

**covariance** : eigenvalues of the covariance matrix of points sampled from the neighborhood area (3x4 descriptors),

**slippage coefficient** which are 6 eigenvalues of the slippage covariance matrix [104] of points sampled within the neighborhood area; we have also included six values being a translational contribution to eigenvectors (12x3 descriptors).

We uniformly sampled the surface of the shapes and computed local descriptors out of those samples. As neighborhood size we have taken 0.01, 0.04, 0.16 and

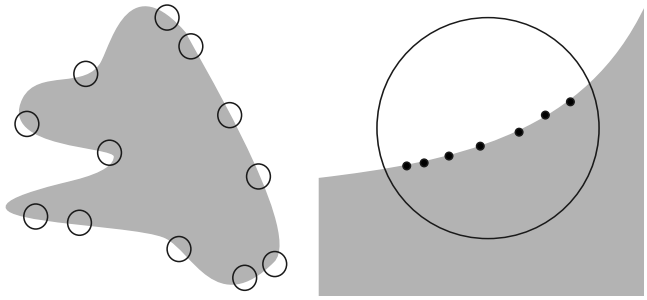


Figure 7.13: Descriptors we use require two levels of sampling. The first level is to take points uniformly sampled on the whole surface (left). Local descriptors computed at those points are collected in the form of a histogram. The second sampling occurs at the level of computing local shape descriptors, when new sample points are taken within a required distance from the base point (right), those points are used to establish measures of that surface, such as curvature which is estimated by fitting primitives into the sampled points.

0.64 of the radius of a bounding sphere of a tooth. For slippage we used 0.01, 0.04 and 0.16.

For each shape we gathered local information coming from any descriptor into a soft histogram, which means that the values are convolved with a Gaussian kernel of a fixed width before being discretized in a histogram. A smooth histogram has the advantage that it induces the smoothness property of a descriptor: if a shape varies smoothly under continuous deformations, the descriptor will also vary continuously [112].

In order to reduce sampling bias we took 2 samplings of 1000 points each, and computed a soft histogram for each of them. Histograms from the two independent samplings were compared. The mean across all training shapes of their difference was taken in order to estimate the measure error coming from the different samplings. Then the mean of the 2 histograms is taken. However in order to compare two histograms for shapes $S_i$ and $S_j$ the distance between two bins is reduced by the previously computed measure error. Then the sum of those values is taken across all bins as our distance $d_k(S_i, S_j)$.

Note that besides those descriptors any descriptors which can compute pairwise distances between shapes can be used here.

## 7.5 Results and Applications

### 7.5.1 Chess Piece Classification

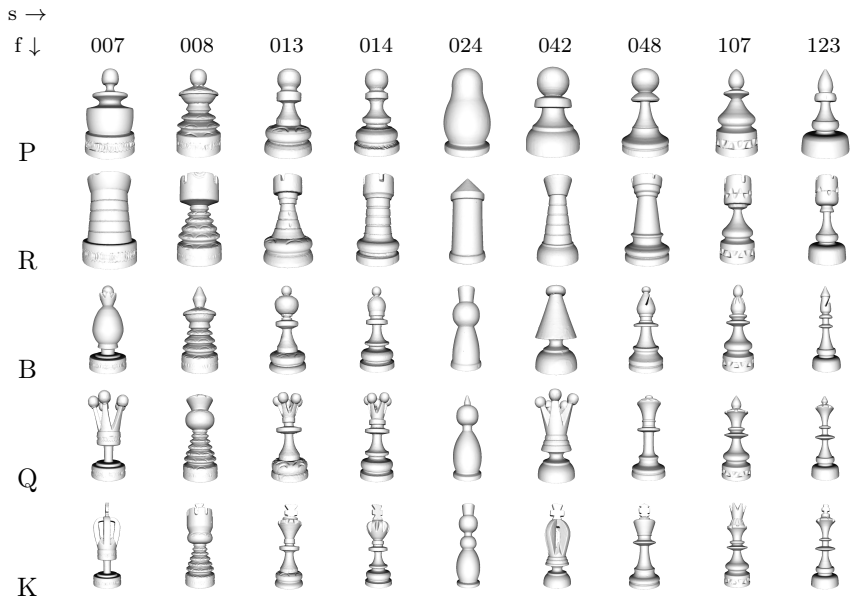



Table 7.1: The chess piece dataset. Our dataset has 45 chess pieces, which are the scans taken from 9 existing chess sets. The function is the type of the chess piece (pawn, rook, bishop, king, queen) and the style is the set the chess piece belongs to.

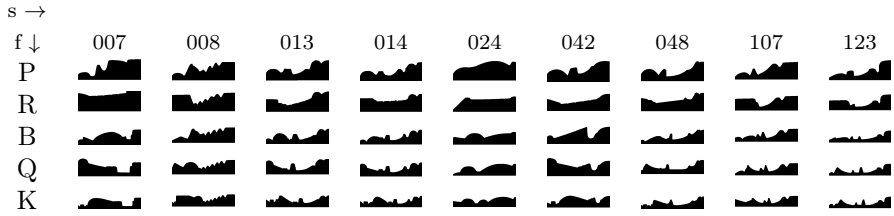



Table 7.2: Outline curves of the chess pieces generated from the three dimensional chess pieces displayed in table 7.1.

In order to obtain the first dataset we scanned 9 different chess sets. The type of the chess set is the style and the function is a chess type. In a standard chess set we have 6 different functions, however we excluded the knight as this piece

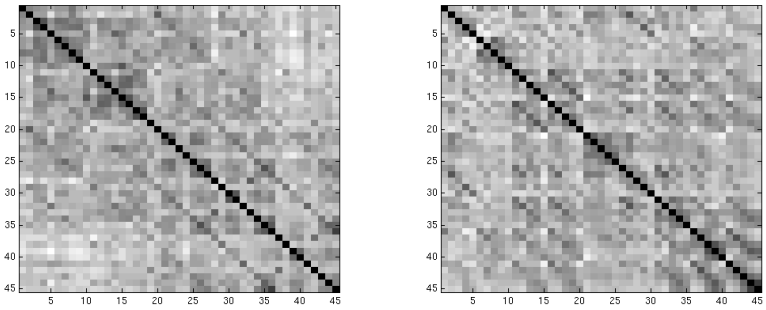



Figure 7.14: Similarities between the curves. Each block has the same function or style, the diagonals of blocks are darker which reflects the smaller distance when the function or the style is the same.

was not rotationally symmetric. This resulted in 5 different functions. Since there is clear rotational symmetry in the chess pieces, their three dimensional representation was reduced to the space of plane curves by taking the outline curve obtained through rotating the chess piece by its rotational symmetry axis (figure 7.2). In order to eliminate the scaling factor each piece was rescaled to the same height. Continuous Translation Invariant Dynamic Time Warping was used in order to establish a similarity metric $d(.,.)$ between the curves (figure 7.14).

In the first experiment we have taken one style and one function as training shapes (refer to section 7.2.2). For the rest of the chess pieces the assignment problem was solved. Table 7.3 contains the result of such assignment where for each style and function we have three values: the first indicates how many test shapes had wrong label, the second how many shapes had wrong function label and third how many had wrong style label. We have also calculated the average performance for all styles and functions. The results depend a lot on the type of the set and function imposed as an example shape. Some of the sets contain a lot of function information but some others do not. The sets 024 and 008 perform the worst. Also the results for the rooks are always worse than for other functions. Note that if we provide a given style as the training set it is used as a definition for the functions and if we give some function it is used as a definition of style. Sometimes we also had a situation when introducing a difficult set into the training data improved the results, because then the labeling of such style was not interfering with the labeling of other styles. So by putting into the training data a set which has a difficult to distinguish style but a clear function or putting a difficult to distinguish function but a clear style distinction usually improved the results.

| | P | R | B | Q | K | mean |
|---|---|---|---|---|---|---|
| 007 | 15 | 23 | 8 | 10 | 21 | 15.4 |
| f | 8 | 12 | 6 | 7 | 12 | 9 |
| s | 14 | 15 | 4 | 6 | 16 | 11 |
| 008 | 22 | 24 | 19 | 16 | 24 | 21 |
| f | 18 | 21 | 15 | 10 | 17 | 16.2 |
| s | 14 | 18 | 9 | 7 | 11 | 11.8 |
| 013 | 14 | 24 | 6 | 6 | 19 | 13.8 |
| f | 7 | 11 | 2 | 2 | 8 | 6 |
| s | 8 | 15 | 4 | 4 | 13 | 8.8 |
| 014 | 20 | 20 | 10 | 9 | 14 | 14.6 |
| f | 6 | 11 | 2 | 2 | 6 | 5.4 |
| s | 17 | 13 | 8 | 7 | 10 | 11 |
| 024 | 28 | 27 | 19 | 25 | 28 | 25.4 |
| f | 20 | 22 | 15 | 20 | 22 | 19.8 |
| s | 16 | 12 | 8 | 12 | 14 | 12.4 |
| 042 | 20 | 24 | 21 | 14 | 18 | 19.4 |
| f | 14 | 14 | 13 | 9 | 10 | 12 |
| s | 11 | 16 | 9 | 7 | 12 | 11 |
| 048 | 16 | 17 | 11 | 10 | 17 | 14.2 |
| f | 7 | 6 | 4 | 2 | 8 | 5.4 |
| s | 14 | 12 | 8 | 8 | 12 | 10.8 |
| 107 | 17 | 20 | 16 | 12 | 17 | 16.4 |
| f | 13 | 12 | 10 | 5 | 9 | 9.8 |
| s | 12 | 15 | 11 | 7 | 10 | 11 |
| 123 | 9 | 24 | 12 | 17 | 18 | 18 |
| f | 13 | 11 | 5 | 9 | 10 | 9.6 |
| s | 13 | 21 | 9 | 10 | 10 | 12.6 |
| mean | 19 | 22.6 | 13.6 | 13.2 | 19.6 | 17.6 |
| f | 11.8 | 13.3 | 8 | 7.3 | 11.3 | 10.4 |
| s | 13.2 | 15.2 | 7.8 | 7.6 | 12 | 11.2 |

Table 7.3: Mismatches of the single assignment problem with one style and one function given. The table contains the general number of pieces with mismatched total label, mismatched function and the mismatched style. The performance depends on the choice of shapes that were used for the definition of the style and the function.

It is also interesting to see how the mismatches looked like. For that reason we have displayed the sorting results for two cases. In table 7.5 we have a situation that swaps were done within style 008 within pawns and one 3-cycle within rooks, however in the table 7.6 a more complicated long cycle running through many styles and functions is present. Note also that some swaps might have a

| | P | R | B | Q | K | mean |
|---|---|---|---|---|---|---|
| 007 | 17 | 25 | 6 | 8 | 11 | 13.4 |
| f | 8 | 9 | 2 | 2 | 6 | 5.4 |
| s | 13 | 19 | 4 | 6 | 5 | 9.4 |
| 008 | 22 | 28 | 23 | 9 | 8 | 18 |
| f | 18 | 22 | 14 | 4 | 7 | 13 |
| s | 13 | 25 | 14 | 6 | 5 | 12.6 |
| 013 | 15 | 21 | 0 | 0 | 19 | 11 |
| f | 8 | 11 | 0 | 0 | 11 | 6 |
| s | 9 | 14 | 0 | 0 | 12 | 7 |
| 014 | 14 | 12 | 5 | 4 | 17 | 10.4 |
| f | 8 | 8 | 0 | 0 | 9 | 5 |
| s | 9 | 9 | 5 | 4 | 11 | 7.6 |
| 024 | 24 | 25 | 18 | 26 | 27 | 24 |
| f | 20 | 18 | 14 | 20 | 20 | 18.4 |
| s | 14 | 18 | 7 | 12 | 11 | 12.4 |
| 042 | 22 | 21 | 15 | 17 | 23 | 19.6 |
| f | 13 | 14 | 10 | 15 | 13 | 13 |
| s | 14 | 15 | 7 | 2 | 16 | 10.8 |
| 048 | 19 | 19 | 2 | 4 | 12 | 11.2 |
| f | 11 | 9 | 2 | 0 | 5 | 5.4 |
| s | 14 | 14 | 0 | 4 | 9 | 8.2 |
| 107 | 15 | 14 | 10 | 5 | 11 | 11 |
| f | 6 | 7 | 6 | 0 | 5 | 4.8 |
| s | 13 | 13 | 5 | 5 | 8 | 8.8 |
| 123 | 19 | 23 | 9 | 7 | 15 | 14.6 |
| f | 11 | 11 | 4 | 2 | 9 | 7.4 |
| s | 10 | 18 | 5 | 5 | 10 | 9.6 |
| mean | 18.6 | 20.9 | 9.8 | 8.9 | 15.9 | 14.8 |
| f | 11.4 | 12.1 | 5.8 | 4.8 | 9.4 | 8.7 |
| s | 12.1 | 16.1 | 5.2 | 4.9 | 9.7 | 9.6 |

Table 7.4: Mismatches of the multiple assignment problem with one style and one function given. The table contains the general number of pieces with the mismatched total label, mismatched function and mismatched style. The improvement occurs usually for tasks where the single assignment solution didn't have too many mismatches, otherwise the performance stays similar to the performance in the case of the one step assignment.

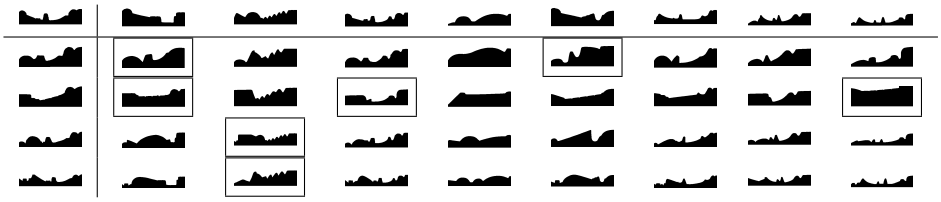


Table 7.5: Example where queens define the styles and 013 define functions. We have one 3 cycle within rooks, a swap between pawns and between pieces from 008 style.

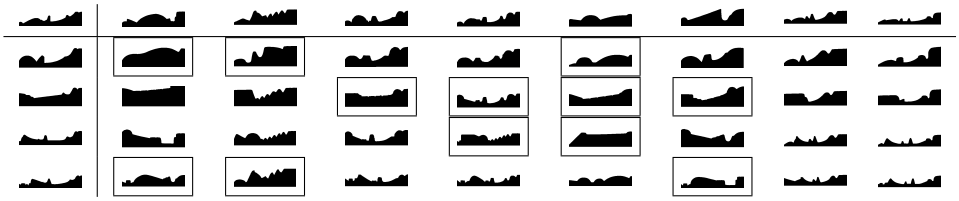


Table 7.6: Example where bishops define the styles and 048 define functions. Note that mismatched assignments can be explained by geometric similarities to the training shapes.

very logical explanation - for example very often something else of a style 008 is labeled as a king instead of the K008. This is because 008 pieces have a kind of collared shape at the top, which might be confused with the upper collared shape of the training king. Note also that bishop 008 as a training shape has attracted shapes P024 and K042, this might be due to the fact that all those three shapes have a rounded barrel shape.

For the multiple step assignment (results: table 7.4, method: refer to section 7.2.2.1) we observe an average improvement of the assignment tasks by approximately 3 chess pieces. Usually if the initial guess is quite good but not perfect then correctness of the matching may be improved quite well. If there are too many mismatches the improvement does not occur: as then we also take as reliable the matchings which are not correct. Usually it does not make the solution worse but keeps it at a similar level as it was with the initial problem.

### 7.5.2 3D Tooth Consistency

A dataset we use for this problems contains teeth shapes (table 7.7) from 6 different mouths. We treat the type of mouth as style and the tooth type as a function. In order to make the number of styles larger, we assume that the left

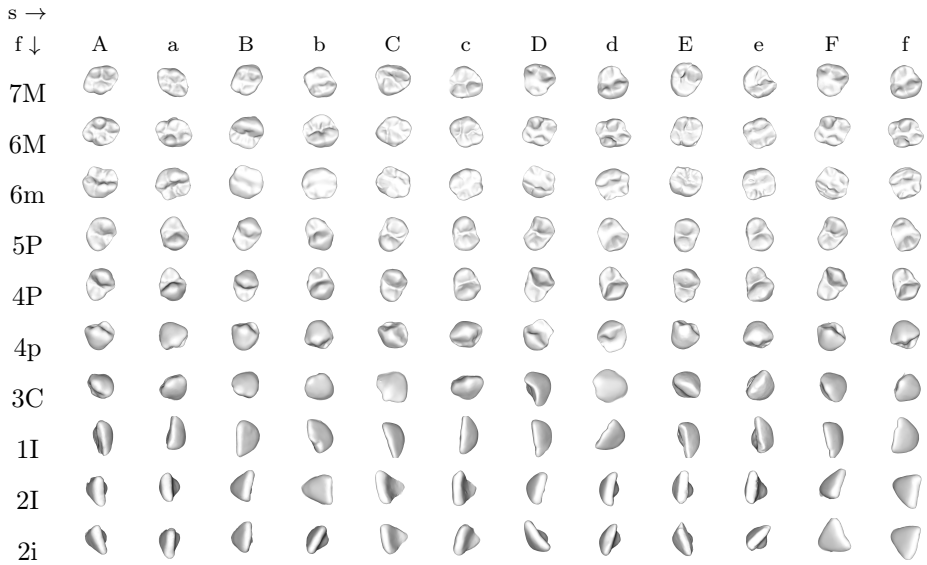



Table 7.7: Tooth dataset

side of a mouth will be treated separately from the right side. This assumption is correct as long as we don't use any descriptors related to symmetry orientation. Thus we have 12 styles which we will label as $A, B, C, D, E, F, a, b, c, d, e, f$, where big letter means one left part of a mouth and small the other one. We have taken 10 tooth types: 2 upper molars, lower molar, 2 upper premolars, lower premolar, upper canine, upper incisor, 2 down incisors. They are labeled and placed in the following order: $7M, 6M, 6m, 5P, 4P, 4p, 3C, 1I, 1i, 2i$, where upper case means the upper tooth.

The descriptors from the section 7.4.2 were used for which the total consistency factors were computed as mentioned in section 7.3.1 and the final dissimilarity measure is created from the measures.

In the first experiment we analyze the metric obtained through the consistency learning process. Because of the size of our dataset we included all of the teeth data. Figure 7.15 contains the resulting dissimilarity measure. What is worth noting is that the similarity between corresponding styles coming from symmetric teeth was clear. During our tests we have discovered that the styles $d, D$ and $f, F$ are very similar and they probably came from the same mouth but are differently meshed. In order to compare the obtained dissimilarity measure with the situation without consistency information, we created the other measure as the average of all available measures (figure 7.16). From the first glance, just

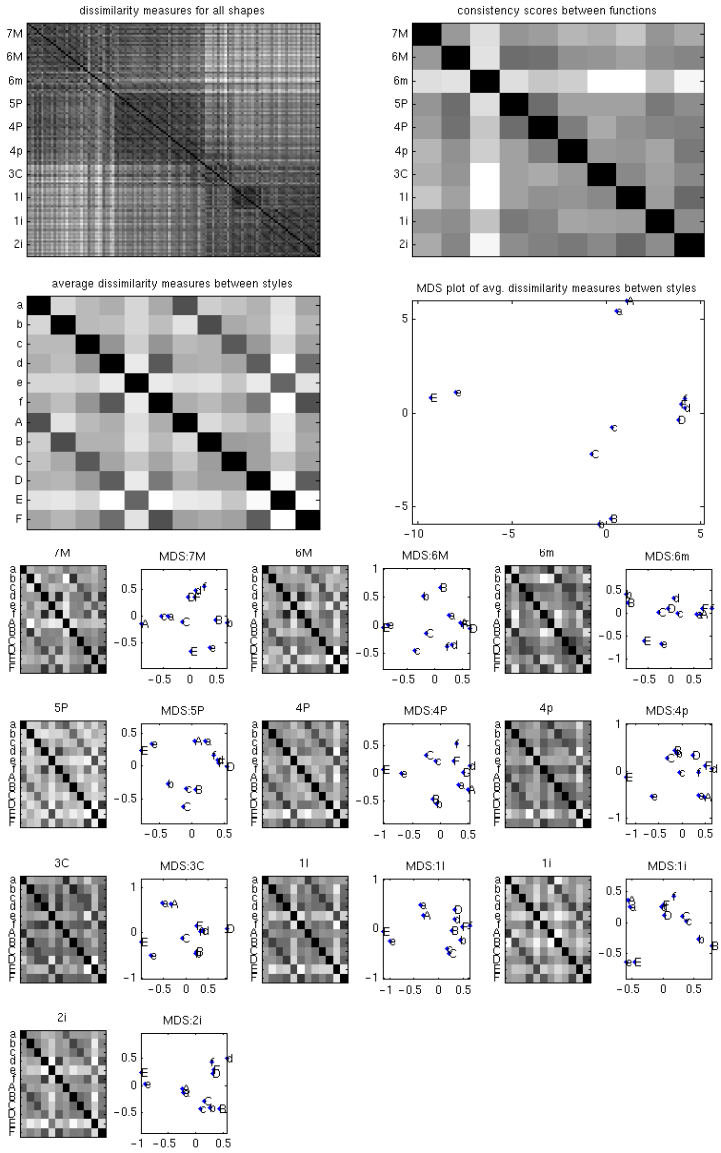



Figure 7.15: Dissimilarity measures obtained with the consistency weights. Note similarities between styles coming from the left and right sides of the mouths (second upper left plot) and low consistency scores (upper right plot) calculated between similar teeth: for example upper neighbors: 6M,5P,4P or upper and lower first premolar (4P and 4p) or the incisors (1I and 2i).

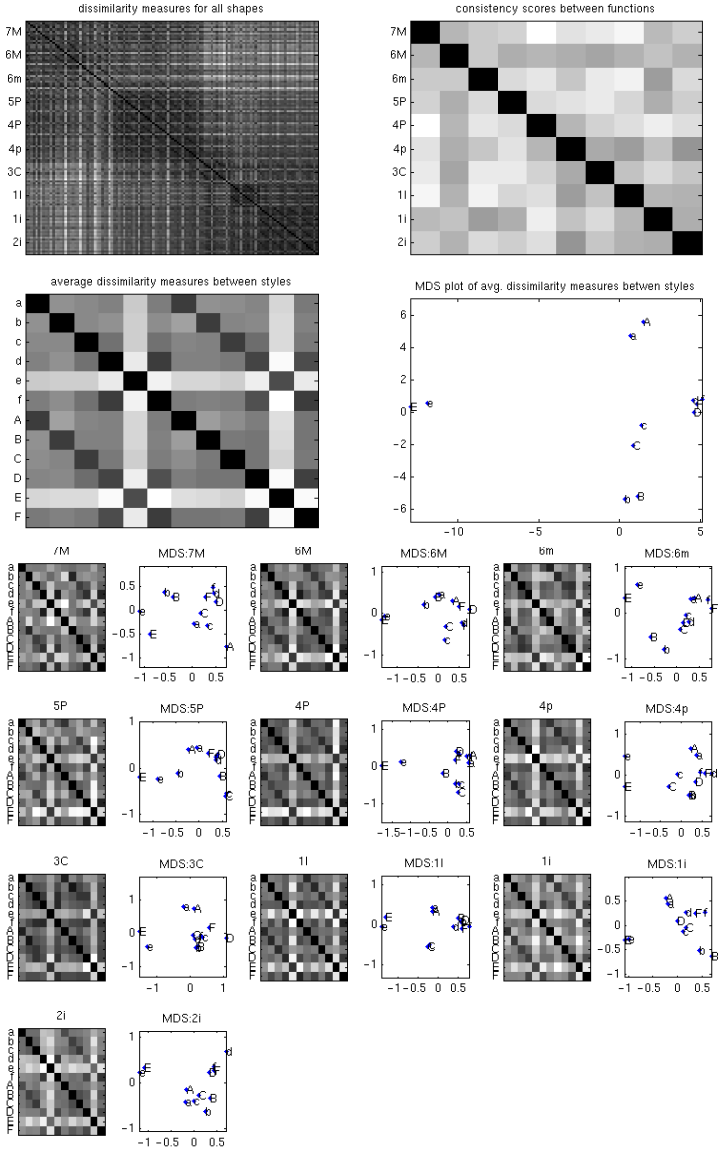



Figure 7.16: Dissimilarity measure obtained with the equal weights. The average structure of differences between styles seem to be similar to the consistency customized measure, besides the tendency of styles $e, E$ to become even more distant than the rest of the shapes. Note however that the consistency scores are very different than the previous example. Also there are higher differences between the lower plots when compared to figure 7.15

when looking at the average distances between styles, the results look very similar no matter if we use the consistency weights or not. However if we look closer and analyze the dissimilarities between the classes functionwise we can see the difference (lower plots of figure 7.15 and figure 7.16). Dissimilarities with consistency weights are more coherent than the average dissimilarities. Also note that the consistency matrix of the average looks more random (upper right plot of 7.15 and 7.16), while the one with the weights has more intuitive information: we can see that the molars are more consistent with each others than incisors, consistency also grows if we have neighbor teeth.

### 7.5.3 Replacing a Missing Tooth

In this section we show an application of the consistency based query method described in section 7.3.2 to a tooth dataset in the tooth replacing scenario.

We start by choosing a training dataset which has several exemplar styles where each style is complete, which means it contains all functions. Then we use the training dataset in all phases of computation when training is needed: to compute the measure error for histogram based descriptors, to compute the total consistency factors $TCF_k$ for given descriptors and finally as a supply to select the best replacement for a missing shape.

We take some style which was not in the training set, and assume that one shape of a function $i$ is missing. In order to replace it we choose from the training database a shape being of function $i$, which we estimate to be the closest to the missing one. Because we don't have the missing shapes we cannot measure the distances directly. Instead we make an estimation based on the distances $D_f(S_k^j, Q_k)$ between the shape of the query style and the functions $k$ that we have and the shapes from the database sharing this function.

In our results we used three strategies of combining $D_f(S_k^j, Q_k)$:

**average** $D^a(s = j, Q)$ distances summed with equal weights (see section 7.3.2),

**weighted** $D^w(s = j, Q)$ distances summed with weights according to consistency factors (see section 7.3.2),

**best** direct use of $D_f(S_k^j, Q_k)$ for function $k$ having the smallest consistency factor $cs_{D_f}^{ki}$ with the function $i$ of a query object.

Note that the 'dissimilarity measure' $D_f$ is used in all cases. As previously mentioned $D_f$ is computed as consistency weighted according to the TCF score

of a given descriptor. What is different here is how we combine the distance if we have more than one function to be checked. The weighted scheme takes distances from all functions with weight according to consistency factors evaluated between the function of an unknown object and the function for which we are computing the weight.

We have also added 'true' which is the distance to the shape we assumed is unknown to the shapes of the same function from the training database. It can be seen as a ground truth. Note however that the distance $D_f$ we use here was obtained by consistency learning as we don't have a ground truth distance.

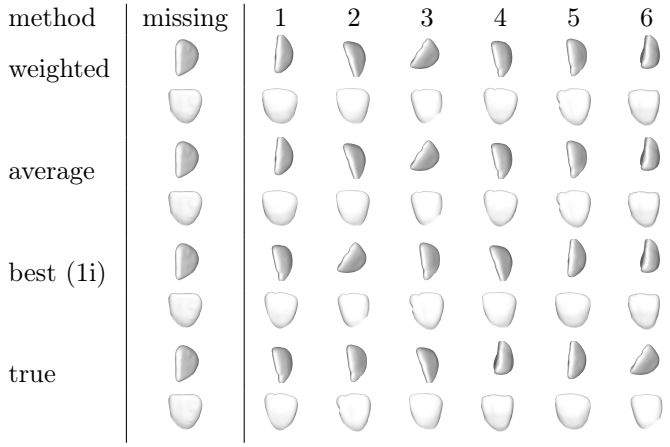



Table 7.8: Replacing missing 1I b shape. For this case the difference between the candidates chosen by our methods is hard to spot visually. This illustrates that even if we get styles with different labels than the ground truth, this does not mean that the alternative choice is bad. The other candidate can also be a good replacement for the missing object.

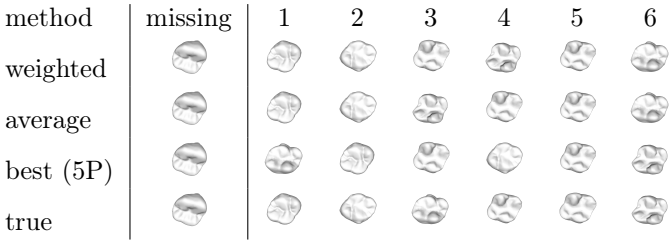



Table 7.9: Replacing missing 6M b. Visual inspection indicates that weights and average strategy seem to choose the most similar tooth - this tooth was also chosen by the ground truth distance.

The results for different queries are shown in the form of labeled plots of dis-

tances. We included results obtained with a training dataset $cdfACD$, and with queries made for all of the functions of styles $a$, $b$, $E$ and $F$ (figures 7.17, 7.18, 7.19 and 7.20 respectively).

From those examples we can see that for our dataset we have some queries which are very easy as for example $a$ (figure 7.17) and always result in finding the symmetric counterpart. When searching for $F$ (figure 7.20) there is label shuffling at the top of the rankings, but all pieces which are assessed to be the closest come from the very similar styles $f, d, D$: either from differently meshed teeth or symmetric counterparts. Thus even though there is a change in the ranking, any of those candidates will do a good job as a replacement. More ambiguous are the queries for which we do not have a symmetric counterpart. As our dataset is not so big this corresponds to the case when we don't have a very close replacement. For such cases we see a tendency that for most of the functions there is a clear solution which is the average replacement that works - as the style $C$, but for some of the tooth types other candidates appear as top on the list: for example $A$ or $f$ when asking for $b$ (figure 7.18) and $A$ or $D$ when asking for $E$ (figure 7.19). Usually the ambiguous cases are among incisors - this might be a good hint to treat the front teeth separately from molars and premolars in order to reach better performance. For the ambiguous cases sometimes the 'best' strategy of replacement is a correct one, but it might also cause overfitting for other cases. The 'average' and 'weighted' strategies have a very similar results. We do not know however if the swaps are a method failure or if this is also the case when one shape might be as good as the other one. This can be left to subjective judgment of the user. To illustrate such case, we have picked two example queries from the figure 7.18: first molar of style (6M) and first incisor (1I) of style to be shown with the images of the ranked replacement candidates (table 7.9 and 7.8). For better insight we have added the front view of incisors. We have also reflected symmetric counterparts of the second part of the mouth (see dataset description in section 7.5.2) for easier spotting of the differences.

We also tested how the consistency properties of the dissimilarity measure change when different subsets of styles were used as the training dataset. We generated a dissimilarity measure from the training data and evaluated the results on all of the data. Usually removing only a small number of mouths does not increase or even slightly decreases the consistency scores. Only when using 3 or 4 mouths, the results seemed to be different. This might come from the fact that there was always some symmetric tooth left in the set which was able to set the consistency scores in a correct way. The increase was mostly noticeable when styles which are close to each other are used as the training set (table 7.10).

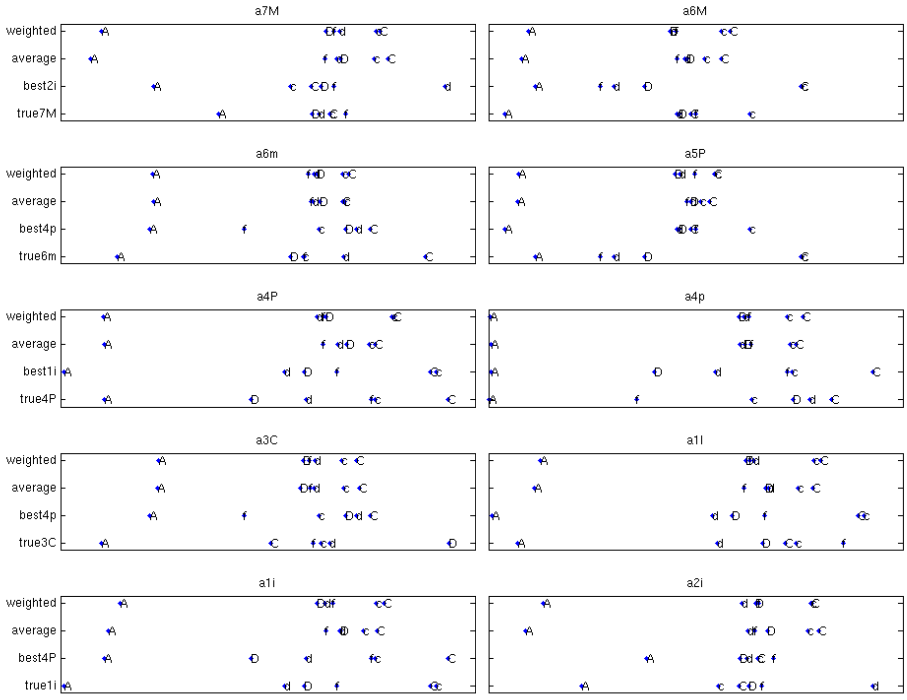



Figure 7.17: Replacing object of a style $a$, we assume we have all the functions of style $a$ except one. Training dataset contains $cdfACD$ styles. This is an easy case as the winner (the shape of the smallest distance), comes from a symmetric counterpart of the missing style.

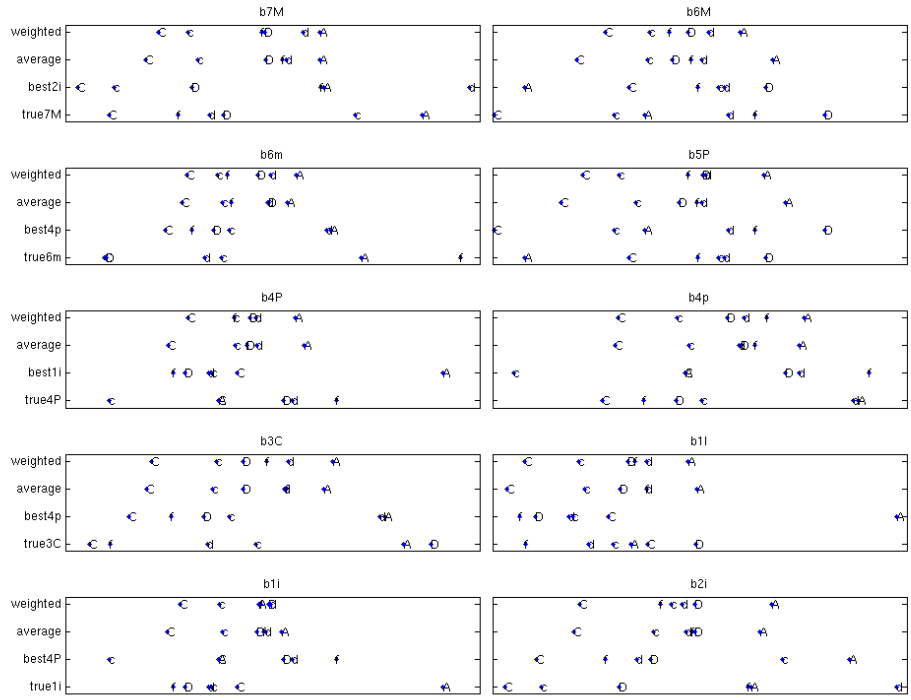



Figure 7.18: Replacing $b$, we assume we have all the functions of style $a$ except one. Training dataset contains $cdfACD$ styles. The average replacement style is $C$ but some exceptions occur especially within the domain of incisors.

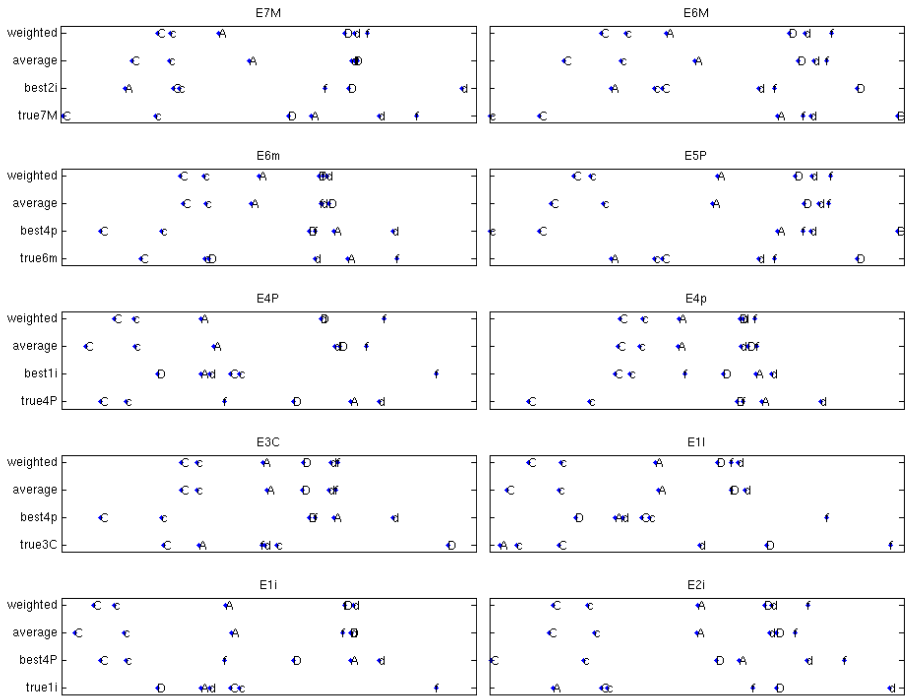



Figure 7.19: Replacing $E$, we assume we have all the functions of style $a$ except one. Training dataset contains $cdfACD$ styles. As in the previous case we have styles $C$ at the top of most rankings with minor exceptions for incisors and function $5p$.

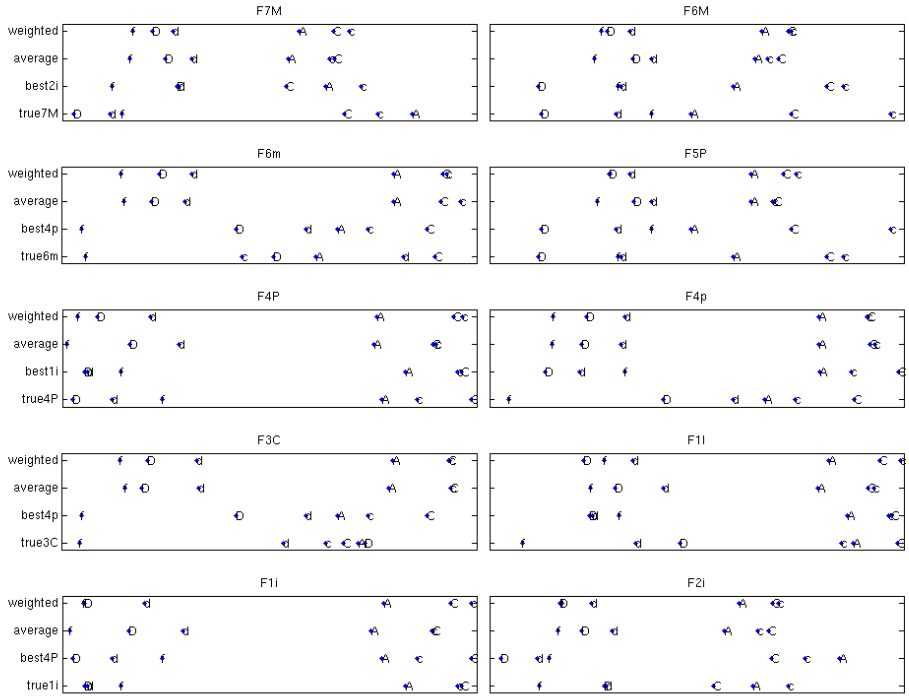



Figure 7.20: Replacing $F$, we assume we have all the functions of style $a$ except one. Training dataset contains $cdfACD$ styles. Although there are some differences at the top of the replacement rankings we can see that all the top candidates come from styles very similar to the query style being $f, d$ or $D$.

| training | TCS | training | TCS | training | TCS |
|---|---|---|---|---|---|
| all | 97.94 | $EFb$ | 102.088 | $cEF$ | 98.17 |
| none | 117.22 | $ABd$ | 99.138 | $ADEFe$ | 97.35 |
| $ABCDFaf$ | 95.929 | $Eef$ | 117.107 | $cdfACD$ | 99.25 |
| $AFade$ | 98.494 | $DdFf$ | 111.007 | $deADEF$ | 115.04 |

Table 7.10: Total consistency factors when using different mouth subsets as training data. First the descriptors were trained by using training data, than all the data was used in order to evaluate TCF. One can see the impact of the choice of training shapes on the quality of the resulting measure. For example using data from very similar styles results in a measure which does not have good consistency for other styles.

### 7.5.4 Chess Pieces Consistency with Comparison To The Warping Distance

As we have 3D representation of chess pieces at our disposal we have applied 3d descriptor computation and the consistency framework, to see the performance of the consistency methods on the chess dataset. We compared the obtained results with the warping distance obtained when using the outline curves (figures 7.21 and 7.22). Some differences occur, however this might be due to the fact that when taking an outline curve a lot of information was lost, so the curve dataset and the style dataset are not totally equivalent. For example the set 008 was very distinguishable in the curve set. However this is not so true for the 3d set, which might be due to the fact that local details at the base are very similar to the details at the base of 007 and 107. Also the upper part has all function characteristic details while when having a curve representation the most dominant was the Christmas-tree-like middle part.

### 7.5.5 Empirical Connection Between Consistency Measure and Dissimilarities Between Styles

As it is much easier to make visual judgments for the dissimilarities between different chess sets than for teeth, we used the chess set example to show the experimentally based connection between the consistency score and dissimilarity measure between different styles. From the 3d images presented in Table 7.1 we assume that some sets can be seen as having standard shapes: 013, 014, 048, 123, 107, while 007 and 042 have the middle part missing and the upper part magnified, 008 is unusual but has a base similar to 007 and 024 - the 'babushka set' can be treated as an outlier.

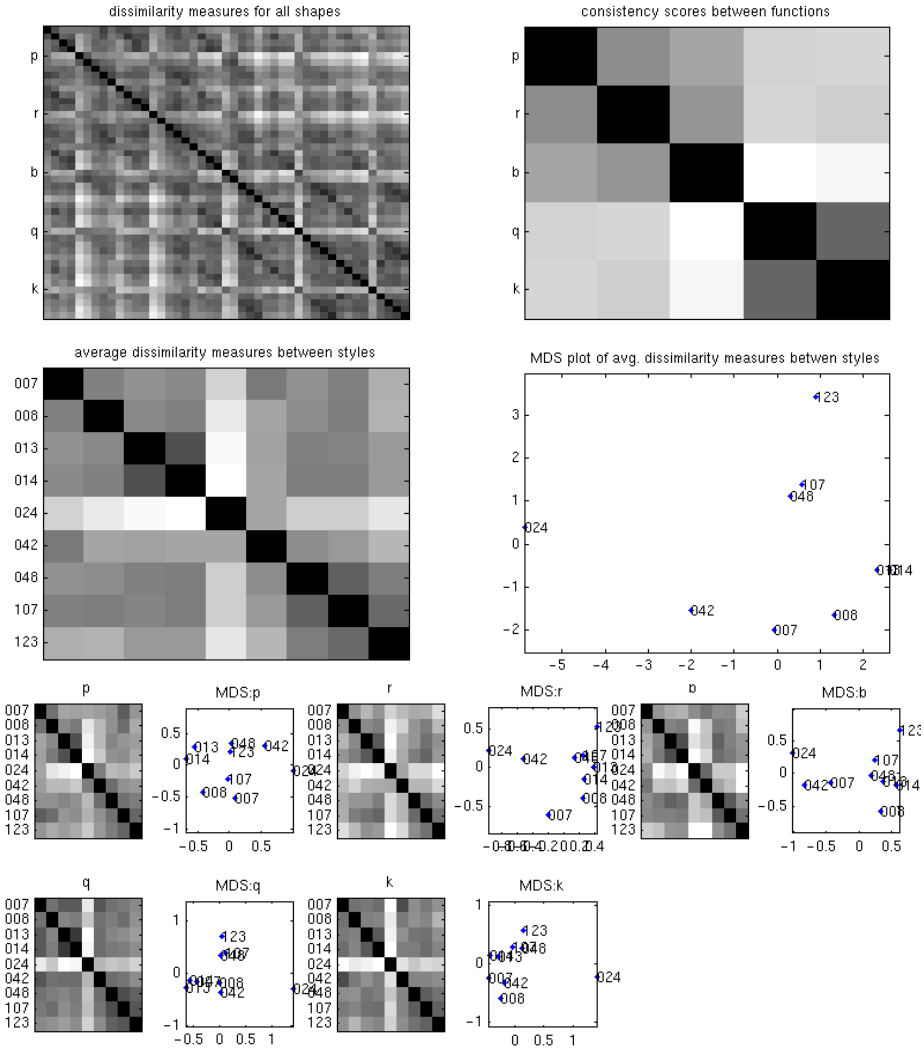



Figure 7.21: Chess piece dissimilarity measure from the 3d descriptors by using consistency weights. The styles are ordered 007,008,013,014,024,042,048,107,123 and the functions are ordered p,r,b,q,k. Note that 008 is not so distinctive as it was for the DTW curve measure. The main source of inconsistency comes from the style 042 which is more similar to the outlier set 024 for pawn, rook and bishop but has more standard shapes for king and queen.

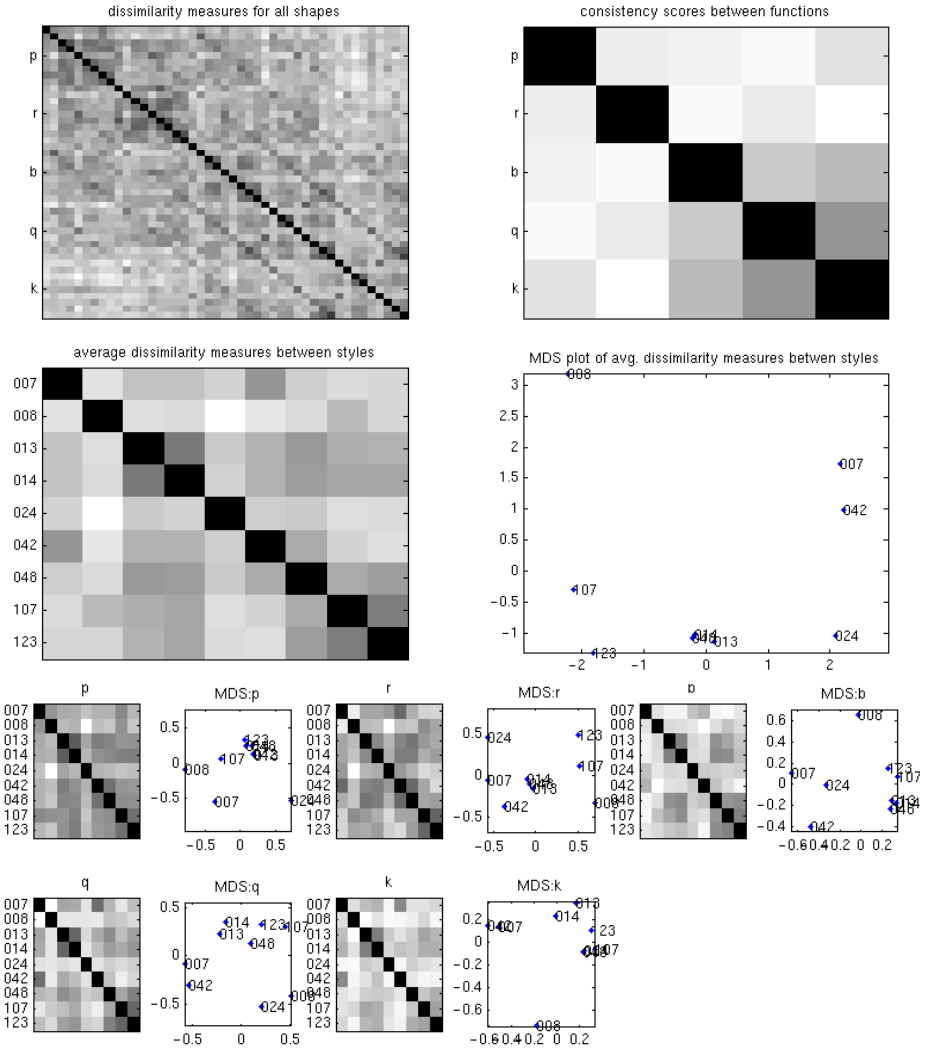



Figure 7.22: Curve warping distance displayed in the same form as consistency based distances. The structure of dissimilarities between styles is similar (with some exceptions like the style 008) to the structure of the dissimilarity coming from consistency weighted 3d descriptors but the general dissimilarity looks slightly different.

From all of our descriptors we have chosen to plot dissimilarity measures for those descriptors which have the best (Figure 7.23), and the worst (Figure 7.24) consistency. We can see that the most consistent descriptor was much better in capturing our judgments, with having standard sets nearby and the non standard 'christmass tree' and 'babushka' as outliers. The least consistent descriptor does not preserve the structure of standard chess pieces being close. For example it treats the set 123 as the most unusual one.

It is worth noting that the TCF value when we used all descriptors combined was 15.19 while the TCF of the best descriptor was 19.35, and of the worst it was 69.27. We can clearly see that combining many descriptors (with weights according to TCF) is better than using just the most consistent descriptor.

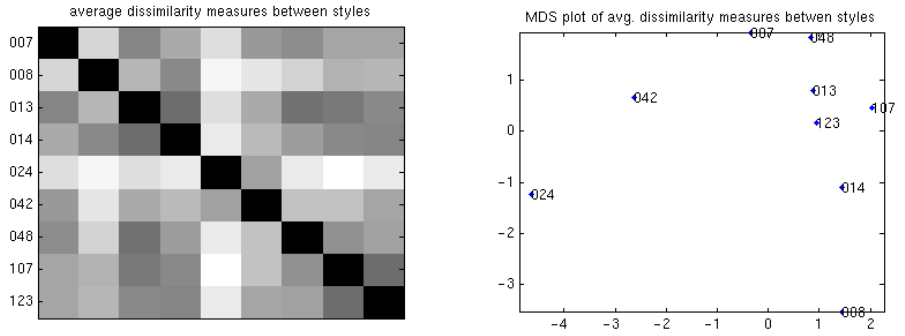



Figure 7.23: Average distances between styles for the descriptor with the best consistency: TCS = 19.35. This descriptor is the histogram of the second eigenvalue of the slippage matrix at scale 0.16.

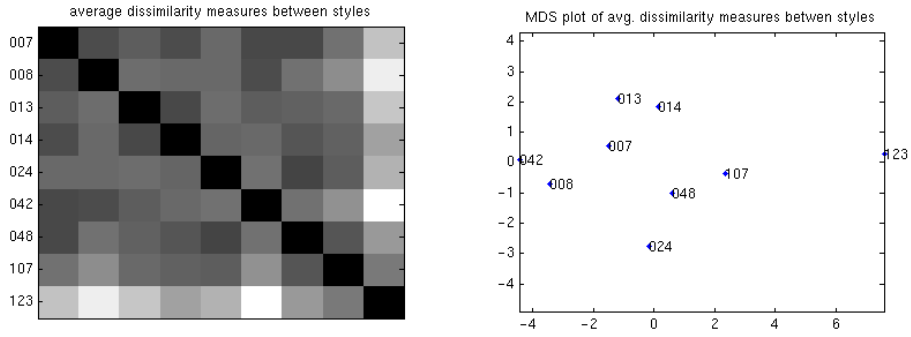



Figure 7.24: Average distances between styles for the descriptor with the worst consistency: TCS=69.27. This descriptor is the histogram of the translational impact of the last eigenvalue of the slippage matrix at scale 0.01

### 7.5.6 Chess Sorting Based on Consistency Measures

In this experiment we sorted the chess pieces with the metric obtained from 3d descriptors from section 7.5.4. Tables 7.11 and 7.12 contain the results for both single and multistep assignment. The results are comparable with the sorting based on consistency measure.

Note that in order to obtain the consistency measures we trained the dataset with all labels known, then we forgot the labels and remembered only the consistency scores. With dissimilarities computed according to the consistency we try to solve the reverse problem - finding the labels back. The sorting and consistency are two different problems, for the consistency we optimize something else - so one does not have a guarantee how well the sorting task will work with the dissimilarity measure obtained by using consistency scores. However, trying to retrieve the labels is an interesting test which measures if consistency based dissimilarity measures are able to store the style and function information that was provided in the training labels. From the sorting results we can see that at least we are not worse than curve matching distance and that the multilevel step outperformed the DTW based distance.

### 7.5.7 Descriptors

The consistency measure process can be see as a black box where we throw the data with a bunch of descriptors and we obtain the consistency customized measure as an output. However it is very interesting to look closer and study the consistency weights in order to see which descriptors had higher impact. For this reason we displayed the style consistency weights for chess and tooth datasets. For both datasets we also exchanged the style and function labels and obtained function consistency weights. All those weights are displayed in figure 7.25. From this plot we see that different descriptors are distinctive for different datasets. Note that for the tooth dataset we have higher weights for smaller scale, for the medium scale the weights get smaller and for bigger scale they grow slightly again. This might be due to the fact that we have organic shapes where their local roughness has importance, but also in the higher scale the general shape counts especially when it comes to determine a function. For the chess pieces the global properties seem to be the more important than the local ones, as for man made objects the global structure of the shape has much importance (such as the existence of big flat or rounded regions or the proportions of such regions).

We can also observe that within one dataset if a descriptor has a high impact on

| | P | R | B | Q | K | mean |
|---|---|---|---|---|---|---|
| 007 | 24 | 17 | 12 | 9 | 20 | 16.4 |
| f | 16 | 10 | 4 | 6 | 8 | 8.8 |
| s | 23 | 15 | 12 | 8 | 18 | 15.2 |
| 008 | 21 | 23 | 20 | 15 | 24 | 20.6 |
| f | 9 | 7 | 10 | 8 | 7 | 8.2 |
| s | 21 | 23 | 19 | 15 | 23 | 20.2 |
| 013 | 19 | 16 | 16 | 3 | 16 | 14 |
| f | 12 | 7 | 6 | 2 | 8 | 7 |
| s | 18 | 16 | 16 | 3 | 14 | 13.4 |
| 014 | 11 | 13 | 12 | 5 | 10 | 10.2 |
| f | 5 | 7 | 2 | 4 | 6 | 4.8 |
| s | 11 | 13 | 12 | 5 | 10 | 10.2 |
| 024 | 25 | 22 | 17 | 12 | 20 | 19.2 |
| f | 10 | 11 | 9 | 6 | 10 | 9.2 |
| s | 25 | 21 | 16 | 12 | 19 | 18.6 |
| 042 | 22 | 23 | 10 | 9 | 17 | 16.2 |
| f | 14 | 13 | 4 | 2 | 6 | 7.8 |
| s | 21 | 22 | 10 | 9 | 16 | 15.6 |
| 048 | 13 | 17 | 16 | 14 | 17 | 15.4 |
| f | 9 | 9 | 7 | 8 | 7 | 8 |
| s | 12 | 17 | 16 | 14 | 16 | 15 |
| 107 | 16 | 13 | 12 | 9 | 15 | 13 |
| f | 11 | 10 | 7 | 7 | 4 | 7.8 |
| s | 15 | 13 | 12 | 9 | 15 | 12.8 |
| 123 | 26 | 17 | 19 | 19 | 19 | 20 |
| f | 15 | 11 | 3 | 11 | 6 | 9.2 |
| s | 25 | 16 | 19 | 19 | 19 | 19.6 |
| mean | 19.67 | 17.89 | 14.89 | 10.56 | 17.56 | 16.11 |
| f | 11.22 | 9.44 | 5.78 | 6 | 6.89 | 7.87 |
| s | 19 | 17.33 | 14.67 | 10.44 | 16.67 | 15.62 |

Table 7.11: Mismatches of the single assignment problem with one style and one function given. The table contains the general number of pieces with mismatched total label, mismatched function and the mismatched style. Results are comparable with the DTW sorting results presented in table 7.3.

| | P | R | B | Q | K | mean |
|---|---|---|---|---|---|---|
| 007 | 25 | 19 | 0 | 0 | 26 | 14 |
| f | 15 | 12 | 0 | 0 | 15 | 8.4 |
| s | 25 | 17 | 0 | 0 | 25 | 13.4 |
| 008 | 10 | 17 | 14 | 4 | 26 | 14.2 |
| f | 4 | 9 | 8 | 4 | 12 | 7.4 |
| s | 10 | 16 | 14 | 3 | 26 | 13.8 |
| 013 | 16 | 13 | 0 | 5 | 12 | 9.2 |
| f | 8 | 2 | 0 | 5 | 4 | 3.8 |
| s | 16 | 13 | 0 | 5 | 12 | 9.2 |
| 014 | 15 | 15 | 0 | 5 | 13 | 9.6 |
| f | 6 | 5 | 0 | 5 | 3 | 3.8 |
| s | 15 | 15 | 0 | 5 | 13 | 9.6 |
| 024 | 22 | 22 | 0 | 5 | 24 | 14.6 |
| f | 14 | 11 | 0 | 5 | 11 | 8.2 |
| s | 22 | 20 | 0 | 4 | 22 | 13.6 |
| 042 | 21 | 19 | 0 | 0 | 14 | 10.8 |
| f | 12 | 13 | 0 | 0 | 5 | 6 |
| s | 19 | 18 | 0 | 0 | 14 | 10.2 |
| 048 | 17 | 15 | 5 | 3 | 18 | 11.6 |
| f | 10 | 7 | 2 | 3 | 9 | 6.2 |
| s | 16 | 14 | 5 | 2 | 17 | 10.8 |
| 107 | 17 | 23 | 0 | 6 | 13 | 11.8 |
| f | 10 | 16 | 0 | 5 | 6 | 7.4 |
| s | 16 | 22 | 0 | 6 | 13 | 11.4 |
| 123 | 18 | 21 | 6 | 2 | 20 | 13.4 |
| f | 10 | 9 | 2 | 2 | 11 | 6.8 |
| s | 18 | 19 | 6 | 2 | 17 | 12.4 |
| mean | 17.89 | 18.22 | 2.78 | 3.33 | 18.44 | 12.13 |
| f | 9.89 | 9.33 | 1.33 | 3.22 | 8.44 | 6.44 |
| s | 17.44 | 17.11 | 2.78 | 3 | 17.67 | 11.6 |

Table 7.12: Mismatches of the single assignment problem with one style and one function given. The table contains the general number of pieces with mismatched total label, mismatched function and the mismatched style. The results are by approximately 2 pieces better than the DTW distance results presented in table 7.4.

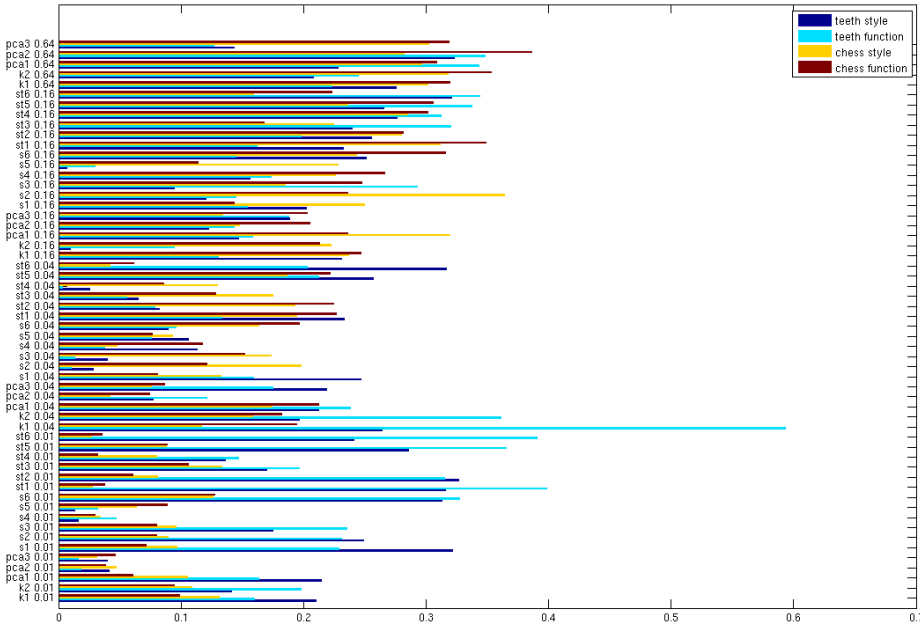



Figure 7.25: Consistency weights used to create the style and function consistent descriptors for tooth and chess datasets. Note that there is larger difference of weights between the datasets than between styles and functions within one dataset.

style (or function) consistency it very often has higher impact on function (or style) of that dataset. This indicates that in such cases we can find descriptors which are appropriate for style and function classification for a given dataset but those descriptors are not purely responsible for style or function but are distinctive for both.

## 7.6 Discussions

In this paper we presented a general framework, which avoids defining style related features explicitly. Instead we introduced a method of finding valuable style related descriptors by applying consistency criteria to the example dataset. We have also shown that even without pointing out the descriptors containing pure style related properties, some practical tasks can be achieved by factoring out the main property of an object, its function. Therefore we do not claim to have discovered a single descriptor or a set of descriptors which are responsible

for style or for function. But we claim that if we have descriptors which at least contain some of the style information, and possibly other stuff, we can still perform well in at least some of the tasks related to the object that have both style and the function. We also confirm the need for treating shapes in broader context than just one shape [106,112] and its geometric descriptors, by analyzing whole sets of shapes from the specific domain.

Appendix A

# Example Based Style Classification

Katarzyna Wełnicka
Jakob Andreas Bærentzen
Henrik Aanæs
Rasmus Larsen

Technical University of Denmark

In Proceedings of the Workshop on Mesh Processing in Medical Image Analysis (in conjunction with MICCAI 2011) Toronto, Canada September 18th, 2011.

Article received the best paper award.

We address the problem of analysis the families of shapes which can be classified according to two categories: the main one corresponding usually to the coarse shape which we call the function and the more subtle one which we call the style. The style and the function both contribute to the overall shape which makes the general analysis and retrieval of such shapes more challenging. Also there is no single way of defining the style as this depends much on the context of the family of shapes used for the analysis. That is why the definition needs to be given through the examples.

The straight forward way of finding the shape descriptors 'responsible' for a given category would be to use well known statistical methods and find through them such descriptors with which we are able to classify shapes according to a given category. When a function is dominating this approach might not suffice - we might be unable to find a set descriptors which are independent of a given function. We show how to decouple the effect of the style from that of the function by considering the shapes of the same function but different styles. We also propose a metric coanalysis approach: if two styles are similar this similarity should be reflected across different functions.

We show the usability of our methods first on the example of a number of chess sets which our method helps sort. Next, we investigate the problem of finding a replacement for a missing tooth given a database of teeth.

## A.1 Introduction

While digital shapes are starting to have a number of medical applications, for instance related to hearing aid production and dental work, the use of digital shapes does not necessarily lead to complete automation. Typically, certain procedures are still left to human operators. However, it is an important goal to be able to help the human operator as much as possible. The particular scenario which we address in this paper is the selection of tooth shapes which can serve as the starting point for digital models of crowns.

There is a lot of work in the shape analysis and especially shape retrieval community with a task of finding the most similar shape to a query one. However, many shapes might be classified not only according to a single category, e.g. as

being a table or a chair, which we will call the function, but also according to the style: A table and a chair of the same style share common geometric properties which are different from the overall shape. The style or the function both interact and contribute to the overall shape of the object. It is not always easy to separate them and point out geometric elements responsible for a function or a style.

The general distinction between the specific shape properties, which tells which ones are responsible for the style and which for the function, is not possible as this depends on a context. That is why we define the style and function through examples.

### A.1.1 Existing Work Related to Style Function Recognition

Style and function separation in the context of man made three dimensional shapes was recently mentioned by Xu et al. [159], where the style of an object is defined by the proportions (anisotropic scaling) of its parts. It seems to be very intuitive and reasonable approach but this does not exhaust the subject. The style might be hidden in details, repetitions of some patterns or some other types of deformation as well. Very often it is hard to define it mathematically although the human brains usually do not have problems in recognizing it.

In many shape processing articles, even if the problem of style is not addressed in an explicit way there are situations where the space of given shapes is broken into two different independent classification systems. In the deformation transfer [141] different kinds of animals can take similar poses in which case it is quite easy to localize them, as the type of animal is described by an intrinsic metric of the shape surface, and the pose is its embedding in three dimensional space. The idea of geometric texture [4] fits within this framework as it aims to separate overall shape from its geometric details. Application of example based priors for surface reconstruction [52, 116] can also be seen as imposing style of the object.

In image processing field Hertzmann et al. [60] presented a method that given three images, an image with style $A$ and function 1, an image with style $B$ and function 1, an image with style $A$ and function 2, created an image with style $B$ and function 2. The same concept was also explored by this group in the field of curve styles [61]. Other related problems can be present when dealing with images of fonts, separating lightning conditions from the scene and distinguishing between the spok languageen and the accent - all of those three cases were examined through bilinear models by Tenenbaum et al. [147].

Tenenbaum's framework requires establishing one to one correspondences of the parts both for the style and with the function - for example fonts are compared through pixels of a bitmap: in general for different types of shapes obtaining such correspondences is usually hard to achieve. Similar corespondances need to be established across the styles for Hertzman's work. Our approach does not require any correspondence finding, which usually is a costly task and sometimes it is not possible as for example in the the task of registeringae table to a Instea d we do shape comparisons through the shape descriptors. There is a lot of current work on content based shape retrieval and different descriptors might capture different properties of the shape. Each andduce a dferent notion of simitheir larity bet. So a good approach is to extract many different shape descriptors and combine them in a proper way.

### A.1.2 Metric Learning

If the feature space is available, many well established statistical methods can be used such as Linear Discriminant Analysis [96] which modifies the feature space so that, for a given training set containing objects from different classes, it maximizes intra class variance and minimizes within class variance. Similar approach was also used by [157] which gives the possibility of defining the similarity and dissimilarity relationships between selected pairs of objects.

As mentioned by Giorgi et al. [57] for the case of shapes there are many useful shape descriptors like skeletons, trees, weighted point sets, which do not provide multidimensional feature space. Still with such descriptors there is usually a way of establishing a notion of similarities between different shapes which results in some kind of pseudodistance.

Giorgi et al. [57] customize a set of distances between of a shape so that user defined similarity is captured. In this work the metric is modified in order to reflect the user defined constraints of nearby or far away shapes. Our work also build a distance through composimaximum distance from distances given by all of the metrics, however the particular metrics are scaled according to a similarity feedback provided by the user.

The approach of combining different metrics relies on the fact that at least there exists a set of shape descriptors which can capture the similarity imposed by virtue of shared stylistic or functional properties. For function, which usually is easier do distinguish such an approach would be very suitable. However when a style needs to be extracted it might not be enough and not even single descriptor might exist which is purely responsible just for the style.

One of our main observations concerning this problem is that knowing what is the function of an object enhances the possibilities for style recognition. For many descriptors information on style is coupled with information on function. In general, when the distance between two shapes is small, it might be both due to similarity in the style and similarity in the function. The retrieval of style related information can be achieved when providing a set of shapes sharing the function and having different styles.

The requirement of recognizing the object of the same function or the same style as being close is not enough in such case. We also want our dissimilarity measures between shapes to be consistent across different functions. This requirement stems from the fact that we want to be able to find the most similar styles and most similar functions. However, for our style-function task case we do not have a direct input which indicates which styles are similar and which are not. Instead we have some notions of similarities which are induced by different shape descriptors and there is a need to chose the ones which are relevant. This relevance is not defined directly by indicating the shapes which should be treated as similar but indirectly as a consistency requirement: dissimilarity or similarity between the styles should be reflected in a similar way for different functions.

Similar indirect consistency approach methodology can be found in [162] which removes incorrect mappings of sets of different views. The assessment of the quality view mappings is done through analyzing them in broader context of the consistent mapping loops. If the loop is inconsistent it means that one of the mappings that belongs to it is wrong and the consistent loop means that mappings are likely to be correct. Having evaluated the correctness of many loops the bad mappings are spotted through a loopy belief propagation.

### A.1.3 Contribution

This paper focuses on an issue, which we think has many application areas, but was not very much explored yet: the analysis and classification of shapes according to more than one category, when categories may be coupled together which in our case is the style and the function.

We propose here a general methodology which can be applied in order to deal with the style-function determination problem. Because the style and the function strongly depend on the context, defining it by providing example shapes seems to be the most general approach.

We show the method for decoupling the effect of the style from that of the

function. By having as a training dataset the shapes of the same function and different styles, we can factor out the function and determine the most likely style of an unknown shape as the closest shape from the set. In an analogous way by using the shapes of the same style but different functions the unknown function may be retrieved.

We realize that the key to success is to find a good metric between the shapes: metric which can capture both stylistic and functional features. Using the example of chess pieces we show what are the desired properties of such a metric (section A.2) and how to decouple the style from the function when only one metric is available.

We also show how to find an appropriate metric by combining the metrics obtained through different shape descriptors (section A.3). Novel in our case is that we do not only use standard similarity notions but also explore the metric consistency approach. The problem is illustrated with the example for a tooth dataset.

After the example of chess pieces, we focus on teeth as an example medical application. Note that our framework is fairly generic. It could be applied to any type of biological surface which exhibits variation due to both style and function.

## A.2 Decoupling Metric

In this section we will show how an information about style and function hidden in the same metric can be decoupled. This is illustrated by the example of style - function classification based on the chess pieces. Since the chess pieces are rotationally symmetric, their three dimensional representation can be reduced to the space of plane curves by taking the outline curve obtained through rotating the chess piece by the rotational symmetry axis. The Translation Invariant Dynamic Time Warping [45] is used in order to establish a similarity metric $d(.,.)$ between the objects.

### A.2.1 Likelihoods Computation

In our setup the proximity of two shapes can be affected by two factors: the similarity of the style and the similarity of the function. Also dissimilarity with respect to one factor, which usually is a style might be more subtle than the

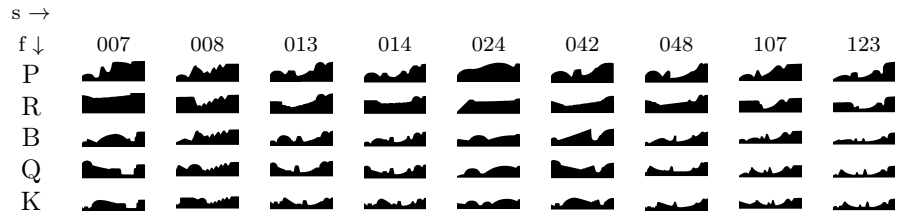



Table A.1: The outline curves of the chess pieces. Our dataset has 45 chess pieces, which are the scans taken from 9 existing chess sets. The function is the type of the chess piece (pawn, rook, bishop, king, queen) and the style is the set the chess piece belongs to.

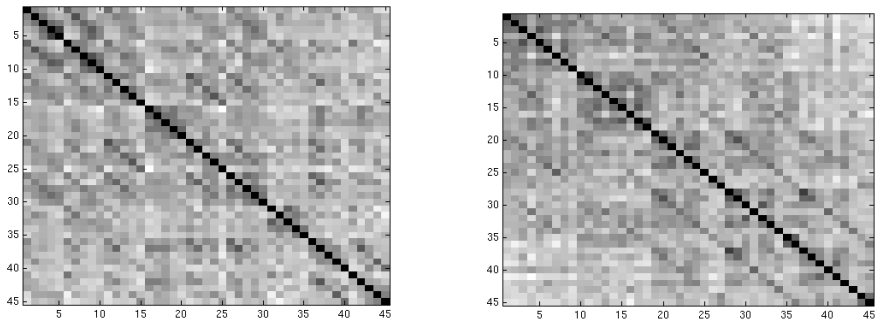


Figure A.1: Similarities between the curves. Each block has the same function or style, the diagonals of blocks are darker which reflects the smaller distance when the function or the style is the same.

other one. However if we have in the training set pieces which share the same style (or function) but have different function (style), then it is possible that we may factor the style (function) out. Instead of taking absolute distances one may use the relative distance information: the difference of the distances. For example if the distance to a king is smaller than a distance to a bishop of the same style we may say that the unknown shape is more likely to be the king than to be something else and that will affect the sign of the distance difference.

The partial likelihood of unknown shape x to be a function $K$, when we have two example shapes of the same style $S_i$ of which one (denoted as $KS_i$) is of a function $K$ and other one $NS_i$ is of a function other than $K$, is equal to:

$$l_f^{S_i,N}(x,K) = d(NS_i,x) - d(KS_i,x).$$

In our training dataset $TS_i$, for a given style $S_i$, we may have more then just

one shape not being of a function $K$ so we take the mean plus the minimum of all of the partial likelihoods:

$$l_f^{S_i}(x,K) = \underset{K \neq NS_i \in TS_i}{\text{mean}} l_f^{S_i,N}(x,K) + \underset{K \neq NS_i \in TS_i}{\min} l_f^{S_i,N}(x,K).$$

Note that minimum is equal to the distance to the closest of the known shapes from style $S_i$ other than $KS_i$, minus the distance to $KS_i$. If the function $K$ is the closest of the shapes from that style, then the minimum will be positive otherwise it will be negative. The mean value stabilizes the results by taking into account distance measures of all of the shapes of this style.

In order to gather the information from all of the training styles we take the mean value plus the maximum of all the styles, for which in a training set there is a function $K$ and some shapes not being of function $K$.

$$l_f(x,K) = \underset{\substack{S_i \in S, KS_i \in TS_i, \\ K \neq NS_i \in TS_i}}{\max} l_f^{S_i}(x,K) + \underset{\substack{S_i \in S, KS_i \in TS_i, \\ K \neq NS_i \in TS_i}}{\text{mean}} l_f^{S_i}(x,K).$$

Here by taking the maximum we are favoring the style for which the K function is most likely. The mean is again added to get the distance information from all known styles.

There might be cases when we do not have enough information in the training set for establishing likelihoods. This happens when there is no set which has a training representative for the function $K$ and for some shape which is not of a function $K$. In such a case we set the likelihood to zero.

The likelihood computation of "x being the style i" is done in an analogous manner. Then for a given x the cost of assigning to it style j and function i is equal to: $l(x,F_i,S_j) = l_f(x,F_i) + l_s(x,S_j)$.

### A.2.2 Chess Classification Example with the Assignment Problem

We use the likelihoods as negative costs and solve the minimum linear assignment problem for the unknown labels and loose chess pieces.

Table A.2 contains the results of the assignment problem if the training dataset is one set and one function, and we are searching for other chess pieces. The results depend a lot on the type of the set and function imposed as an example shape. Some of the sets contain a lot style and function information but some other do not. The sets 024 and 008 are performing the worst also the results for the rooks is always worse than for other functions.

| | P | R | B | Q | K | mean |
|---|---|---|---|---|---|---|
| 123 | 9 | 24 | 12 | 17 | 18 | 18 |
| f | 13 | 11 | 5 | 9 | 10 | 9.6 |
| s | 13 | 21 | 9 | 10 | 10 | 12.6 |
| 107 | 17 | 20 | 16 | 12 | 17 | 16.4 |
| f | 13 | 12 | 10 | 5 | 9 | 9.8 |
| s | 12 | 15 | 11 | 7 | 10 | 11 |
| 014 | 20 | 20 | 10 | 9 | 14 | 14.6 |
| f | 6 | 11 | 2 | 2 | 6 | 5.4 |
| s | 17 | 13 | 8 | 7 | 10 | 11 |
| 024 | 28 | 27 | 19 | 25 | 28 | 25.4 |
| f | 20 | 22 | 15 | 20 | 22 | 19.8 |
| s | 16 | 12 | 8 | 12 | 14 | 12.4 |
| 013 | 14 | 24 | 6 | 6 | 19 | 13.8 |
| f | 7 | 11 | 2 | 2 | 8 | 6 |
| s | 8 | 15 | 4 | 4 | 13 | 8.8 |
| 048 | 16 | 17 | 11 | 10 | 17 | 14.2 |
| f | 7 | 6 | 4 | 2 | 8 | 5.4 |
| s | 14 | 12 | 8 | 8 | 12 | 10.8 |
| 008 | 22 | 24 | 19 | 16 | 24 | 21 |
| f | 18 | 21 | 15 | 10 | 17 | 16.2 |
| s | 14 | 18 | 9 | 7 | 11 | 11.8 |
| 042 | 20 | 24 | 21 | 14 | 18 | 19.4 |
| f | 14 | 14 | 13 | 9 | 10 | 12 |
| s | 11 | 16 | 9 | 7 | 12 | 11 |
| 007 | 15 | 23 | 8 | 10 | 21 | 15.4 |
| f | 8 | 12 | 6 | 7 | 12 | 9 |
| s | 14 | 15 | 4 | 6 | 16 | 11 |
| mean | 19 | 22.56 | 13.56 | 13.22 | 19.56 | 17.58 |
| f | 11.78 | 13.33 | 8 | 7.33 | 11.33 | 10.36 |
| s | 13.22 | 15.22 | 7.78 | 7.56 | 12 | 11.16 |

Table A.2: Mismatches of the single assignment problem with one style and one function given. The table contains the general number of pieces with mismatched total label, mismatched function and the mismatched style.

### A.2.3 Multiple Step Assignment

An assignment problem with the costs defined above does not make use of the information about all of the distances between the shapes. If we are able to locate the chess pieces of which we can expect that the initial matching went correctly we can add those into a training dataset with the labels obtained by the initial assignment. In order to estimate the labeling reliability, we calculate the diagonal cost of the assignment which we define as the average sum of similarities between all the pieces having the same style or function labels. For a hypothetical unknown chess piece we might add it for a moment to the training set and calculate what is the diagonal cost when assignment is solved with the use of this piece. We discovered that instead of calculating diagonal costs directly it is better do the inverse assignment, which is performed by swapping the unknown data with the known and then calculating the diagonal cost. Then the smaller the inverse diagonal cost is the more reliable is the hypothetical assignment of the unknown chess piece to its label.

In order to minimize bad choices we always take the piece having minimum inverse diagonal cost and is reliable according to the additional reliability criteria. We add it to the initial dataset and repeat the assignment and the most reliable pieces addition until there is no reliable piece to be added. Then we use the assignment from the last step as the final assignment.

In the results (table A.3) we observe an average improvement of the assignment tasks by approximately 3 chess pieces. Usually if initial guess is quite good but not perfect then correctness of the matching may be improved quite well. If there are too many mismatches the improvement does not occur: as then we also take as reliable the matchings which are not correct. Usually it does not make the solution worse but keeps it at a similar level as it was with the initial problem.

## A.3 Finding the Good Metric

The case of chess pieces was special problem as we were able to reduce the shape information to the space of the curves and had a way of establishing similarity between those curves by using Translation Invariant Dynamic Time Warping. In general for three dimensional shapes we do not know a good metric in advance, instead we have many propositions of metrics $d_i(,)$ which can be obtained through different kinds of shape descriptors $D_i$.

| | P | R | B | Q | K | mean |
|---|---|---|---|---|---|---|
| 123 | 19 | 23 | 9 | 7 | 15 | 14.6 |
| f | 11 | 11 | 4 | 2 | 9 | 7.4 |
| s | 10 | 18 | 5 | 5 | 10 | 9.6 |
| 107 | 15 | 14 | 10 | 5 | 11 | 11 |
| f | 6 | 7 | 6 | 0 | 5 | 4.8 |
| s | 13 | 13 | 5 | 5 | 8 | 8.8 |
| 014 | 14 | 12 | 5 | 4 | 17 | 10.4 |
| f | 8 | 8 | 0 | 0 | 9 | 5 |
| s | 9 | 9 | 5 | 4 | 11 | 7.6 |
| 024 | 24 | 25 | 18 | 26 | 27 | 24 |
| f | 20 | 18 | 14 | 20 | 20 | 18.4 |
| s | 14 | 18 | 7 | 12 | 11 | 12.4 |
| 013 | 15 | 21 | 0 | 0 | 19 | 11 |
| f | 8 | 11 | 0 | 0 | 11 | 6 |
| s | 9 | 14 | 0 | 0 | 12 | 7 |
| 048 | 19 | 19 | 2 | 4 | 12 | 11.2 |
| f | 11 | 9 | 2 | 0 | 5 | 5.4 |
| s | 14 | 14 | 0 | 4 | 9 | 8.2 |
| 008 | 22 | 28 | 23 | 9 | 8 | 18 |
| f | 18 | 22 | 14 | 4 | 7 | 13 |
| s | 13 | 25 | 14 | 6 | 5 | 12.6 |
| 042 | 22 | 21 | 15 | 17 | 23 | 19.6 |
| f | 13 | 14 | 10 | 15 | 13 | 13 |
| s | 14 | 15 | 7 | 2 | 16 | 10.8 |
| 007 | 17 | 25 | 6 | 8 | 11 | 13.4 |
| f | 8 | 9 | 2 | 2 | 6 | 5.4 |
| s | 13 | 19 | 4 | 6 | 5 | 9.4 |
| mean | 18.56 | 20.89 | 9.78 | 8.89 | 15.89 | 14.8 |
| f | 11.44 | 12.11 | 5.78 | 4.78 | 9.44 | 8.71 |
| s | 12.11 | 16.11 | 5.22 | 4.89 | 9.67 | 9.6 |

Table A.3: Mismatches of the multiple assignment problem with one style and one function given. The table contains the general number of pieces with the mismatched total label, mismatched function and mismatched style.

The task is to choose such a metric $d_i$ or some combinations of metrics with which we can distinguish between different styles. As mentioned in the introduction it is not enough to be able for the objects of the same style to be close but also the dissimilarity measures should be consistent across different functions and we don't know which one will work best for specific problem.

This requirement can be illustrated with the problem of tooth shapes. Suppose a patient has one tooth destroyed. In order to be able to reproduce its shape, we want to find from a database a tooth which is mostly similar to the existing tooth he has. We have a molar missing but because a premolar is still in the patient's mouth, we wish to search in our database for a mouth which has the most similar premolar to the patient's. From that mouth we take a molar as a template for our new tooth. This approach assumes that similarity for premolars induces a similarity between molars.

This case shows that the metric consistency requirement is necessary as it aids in many concrete tasks - like searching for the closest to missing data. Here we do not know directly what 'close' means, as we have many metric but don't know which one is a correct. Usually a correct metric combination in such a case can be found by giving example pairs of shapes which are similar and which are dissimilar [57]. In our case we do not have such information. Instead we can impose the metric consistency requirement: the distances between shapes having different styles and function $A$ should be close to the distances of the shapes of the same styles and function $B$.

### A.3.1 Metric Consistency

Assume we have a set of training shapes $F_{i=1..n_i}S_{j=1..n_j}$, where $i$ indicates the function and $j$ style. We also have a $k_n$ potential distances $d_k(,)$

Let us take all distances $d_k(F_{i_1}S_{j_a=1..n_j}, F_{i_1}S_{j_b=1..n_j, j_b \neq j_a})$ between different shapes of the function $i_1$. In order to be comparable those distances need to be normalized which we do by dividing them by the median from obtained distances. This results in a $\binom{n_j}{2}$ dimensional vector of k-distances between shapes with function $i_1$ which we will denote $v(d_k, f_{i_1})$.

For each pair $i_1 \neq i_2$ of two different functions we can establish the **consistency score** $cs_{d_k}^{i_1,i_2}$ with respect to a distance $k$ and function $i_1$ and $i_2$ as a norm of difference of distance vectors:

$$cs_{d_k}^{i_1,i_2} = \sqrt{\sum_{l=1..\binom{n_i}{2}} (v(d_k, f_{i_1})_l - v(d_k, f_{i_2})_l)^2}$$

Figure A.2: Front view of molar, premolar and incisor from 3 different mouths

In order to calculate total consistency factor ($TCF_k$) for a distance measure $k$ sum of the differences for all function pairs is taken. Note that smaller $TCF_k$ is the more consistent is $d_k$ with respect to style.

We construct the final metrics by summing the metric obtained through different shape descriptors with weights that promote consistency.

$$D_f(,) = \sum_k e^{(-2\frac{TCF_k}{mean(TCF)})} \frac{d_k(,)}{\sigma_{d_k}}$$

where $\sigma_{d_k}$ median distance from distances $d_k(,)$ between all training shapes.

From the final metrics we can also compute consistency scores $cs_{D_f}^{i_1,i_2}$. This consistency measures can be used in order to asses what kind of tooth types are better correlated. For example two neighbor upper molars can be more correlated than molar and incisor. So if a molar is missing and we have the neighbor molar and incisor, we should give higher weight for query of closest mouth with respect to a molar than with respect to incisor. We can also compute mean distances between styles by summing $v(D_f, f_j)$ for all function types $j$.

### A.3.2 The Tooth Problem

In the teeth analysis task we take a type of a mouth as style and a tooth type as function. An example dataset we use for this problems contains teeth shapes (figure A.2) from 6 different mouths. In order to make number of styles larger, we assume that the left side of a mouth will be treated separately from the right part. Thus we have 12 styles which we will label as A,B,C,D,E,F,a,b,c,d,e,f, where big letter means one left part of a mouth and small the other one. We have taken 10 tooth types 2 upper molars, lower molar, 2 upper premolars, lower premolar, upper canine, upper incisor, 2 down incisors. They are labeled and placed in the following order: 7M,6M,6m,5P,4P,4p,3C,1I,1i,2i, where upper case means respectively upper tooth.

In order to get independence of meshing we uniformly sampled the surface of teeth and computed descriptors out of those samples. We used local shape descriptors which rely on neighborhood at some distance from a given position.

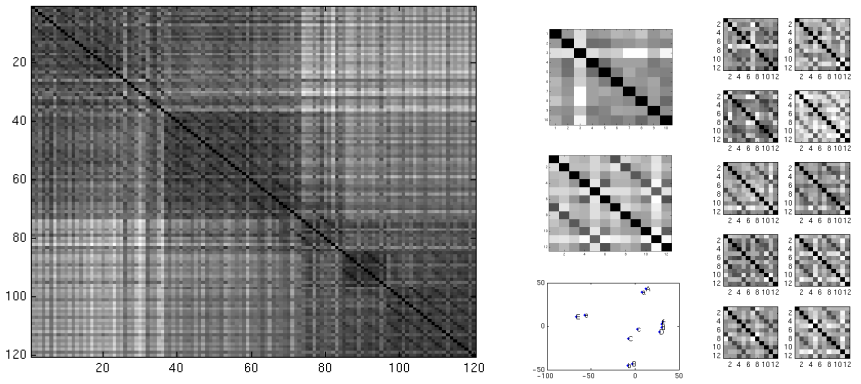



Figure A.3: Left: final metric $D_f$ obtained with teeth database (indices grouped with respect to teeth type). Center: $cs^{i,j}$ for different teeth types, the average distance between the styles, and the multidimensional scaling plot for the avarage distances. Right: metrics between different styles when a function is fixed, obtained form final metric with the training styles $AFade$.

As neighborhood size we have taken 0.01 0.04 0.16 and 0.64 of the radius of a bounding sphere of a tooth. For slippage we used 0.01 0.04 and 0.16. In Total we had: 2x4 descriptors for main curvatures obtained by fitting primitives [146], 3x4 eigenvalues of covariance matrix of points sampled from the neighborhood area and 12x3 slippage coefficients [104] which are 6 eigenvalues of slippage covariance matrix and 6 is a translational contribution to its eigenvectors. We took 2 samples for 1000 points, for which soft histograms were computed. Histograms from two independent sampling were compared. The mean across all training shapes, of their difference was taken in order to estimate the measure error coming from different samplings. Then the mean of the 2 sample histogram is taken. However in order to compare two histograms for shapes $S_i$ and $S_j$ the distance between two bins is reduced by the previously computed measure error. Then the sum of those values is taken across all bins as our distance $d_k(S_i, S_j)$.

Then the total consistency factors are computed as mentioned in the previous section and the final metric is computed. Figure A.3 contains the resulting metric, where all of the available teeth were used. It is worth mentioning that the consistency score for a resulting metric is smaller than the scores from any particular metrics. Note that the styles that come from the same mouth (left or right part) are being found as close. Also note that neighbor teeth tend to have more consistent scores. This information might be used when searching for a missing tooth. Let us consider the case when $AFade$ styles were taken as

| training | TCS | training | TCS | training | TCS |
|---|---|---|---|---|---|
| $DdFf$ | 111.007 | $EFb$ | 102.088 | $cEF$ | 98.17 |
| $Eef$ | 117.107 | $ABd$ | 99.138 | $ADEFe$ | 97.35 |
| all | 97.94 | $AFade$ | 98.494 | | |
| none | 117.22 | $ABCDFaf$ | 95.929 | | |

Table A.4: Total consistency factors when using different mouth subsets as training data.

training styles and a metric $T$ was created. Then a patient comes with mouth of style $C$ and with missing $4P$. We have scans of his $4p$, $5P$ and $1i$ and we have $cs_T^{4P,4p} = 1.6391$, $cs_T^{4P,5P} = 1.5819$ and $cs_T^{4P,1i} = 1.7543$, so we use tooth $5P$ as it has the best consistency. We use $v(T, f_{5P})$ instead of unknown $v(T, f_{4P})$ in order to evaluate proximity between teeth (5th and 6th plot on the Right of figure A.3). We evaluate distances between $C$ and $AFade$ among teeth of type $5P$ and the mouths sorted with respect to distance will be $FdaAE$ if we checked the ground truth we have $dFaEA$. Despite the swaps which was a result of a very close similarity values of $dF$ and $EA$ we can see that in general dissimilar teeth remain dissimilar.

We also tested on how the consistency properties of metric change when different subset of styles was used as training dataset. We generated metric from this information and evaluated the results on all of the data.

Usually removing only small number of mouths did not increase or even slightly decrease the consistency scores. Only when using 3 or 4 mouths, the results seemed be different. This might come from the fact that there was always some symmetric tooth left in the set which was able to set the consistency scores in a correct way. The increase was mostly noticeable when styles which are close to each other are used as training set (table A.4).

## A.4 Conclusion

In this article we presented methods of working with shapes that can be classified into having two categories: style and function. One of them decouples style and function when they are incorporated into the same metric. The second finds a metric as a combination from existing ones when a consistency between different function types is needed. Those methods were illustrated by the chess and tooth datasets. We are aware that for a further analysis and development of our methods more data will be needed but we think the results obtained so far are promising.

#### A.4.0.1 Acknowledgments:

We would like to thank 3Shape for the tooth dataset and for making their scanner available when scanning the chess pieces.

# Bibliography


[1] Ceyhun Burak Akgül, Bülent Sankur, Yücel Yemez, and Francis Schmitt. 3d model retrieval using probability density-based shape descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1117–1133, 2009.

[2] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2-3):105–114, 2003.

[3] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, SMA '01, pages 249–266, New York, NY, USA, 2001. ACM.

[4] Vedrana Andersen. Height and tilt geometric texture, 2009. Presented at: Danish Conference on Pattern Recognition and Image Analysis ; 17 : Ven, 2009.

[5] M. Ankerst, G. Kastenmüller, H. P. Kriegel, and T. Seidl. 3D shape histograms for similarity search and classification in spatial databases. *Advances in Spatial Databases, 6th International Symposium, SSD'99*, 1651:207–228, 1999.

[6] Marco Attene, Silvia Biasotti, Michela Mortara, Giuseppe Patanè, Michela Spagnuolo, and Bianca Falcidieno. Computational methods for understanding 3d shapes. *Computers & Graphics*, 30(3):323–333, 2006.

[7] Marco Attene, Silvia Biasotti, and Michela Spagnuolo. Shape understanding by contour-driven retiling. *The Visual Computer*, 19(2-3):127–138, 2003.

[8] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[9] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

[10] Grégoire Aujay, Franck Hétroy, Francis Lazarus, and Christine Depraz. Harmonic skeleton for realistic character animation. In Dimitris Metaxas and Jovan Popović, editors, *Symposium on Computer Animation, SCA 07, August, 2007*, pages 151–160, San Diego, California, Etats-Unis, August 2007. ACM-Siggraph/Eurographics.

[11] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2002.

[12] Mikhail Belkin, Jian Sun, and Yusu Wang. Discrete laplace operator on meshed surfaces. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 278–287, New York, NY, USA, 2008. ACM.

[13] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[14] Mirela Ben-Chen, Adrian Butscher, Justin Solomon, and Leonidas J. Guibas. On discrete killing vector fields and patterns on surfaces. *Comput. Graph. Forum*, pages 1701–1711, 2010.

[15] Mirela Ben-Chen and Craig Gotsman. Characterizing shape using conformal factors. In Stavros J. Perantonis, Nickolas S. Sapidis, Michela Spagnuolo, and Daniel Thalmann, editors, *3DOR*, pages 1–8. Eurographics Association, 2008.

[16] Marcel Berger. *A panoramic view of Riemannian geometry.* Springer, 2003.

[17] S. Biasotti, L. De Floriani, B. Falcidieno, P. Frosini, D. Giorgi, C. Landi, L. Papaleo, and M. Spagnuolo. Describing shapes by geometrical-topological properties of real functions. *ACM Comput. Surv.*, 40:12:1–12:87, October 2008.

[18] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Size functions for comparing 3d models. *Pattern Recogn.*, 41(9):2855–2873, 2008.

[19] S. Biasotti and M. Spagnuolo. M.: Shape understanding by contour driven retiling. *The Visual Computer*, 19:2–3, 2003.

[20] Silvia Biasotti. Shape comparison through mutual distances of real functions. In *Proceedings of the ACM workshop on 3D object retrieval*, 3DOR '10, pages 33–38, New York, NY, USA, 2010. ACM.

[21] Silvia Biasotti, Bianca Falcidieno, and Michela Spagnuolo. Extended reeb graphs for surface understanding and description. In *DGCI '00: Proceedings of the 9th International Conference on Discrete Geometry for Computer Imagery*, pages 185–197, London, UK, 2000. Springer-Verlag.

[22] H. Blum. An associative machine for dealing with the visual field and some of its biological implications. In E. E. Bernard and M. R. Kare, editors, *Biological Prototypes and Synthetic Systems*, volume 1, pages 244–260, NY, 1962. Plenum Press. Proc "$2^{nd}$ Annual Bionics Symposium, held at Cornell University, 1961.".

[23] M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, and A. Schilling. Symmetry detection using feature lines. *Computer Graphics Forum*, 28(2):697–706, April 2009.

[24] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Br uno Levy. *Polygon Mesh Processing*. AK Peters, 2010.

[25] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[26] Benjamin Bustos, Tobias Schreck, Michael Walter, Juan Barrios, Matthias Schaefer, and Daniel Keim. Improving 3D similarity search by enhancing and combining 3D descriptors. *Multimedia Tools and Applications*, pages 1–28, January 2011.

[27] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas Guibas. Persistence barcodes for shapes. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 124–135, New York, NY, USA, 2004. ACM.

[28] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 177–187, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[29] A. Cerri, D. Giorgi, P. Musé, F. Sur, and F. Tomassini. Shape recognition via an a contrario model for size functions. In *Proceedings of the International Conference on Image Analysis and Registration (ICIAR)*, volume LNCS 4142, pages 410–421, Povoa de Varzim (Portugal), September 2006.

[30] Isaac Chavel, Burton Randol, and Jozef Dodziuk. *Eigenvalues in Riemannian geometry*. Acad. Press, 1984.

[31] Frédéric Chazal and André Lieutier. The "λ-medial axis". *Graph. Models*, 67:304–331, July 2005.

[32] Ding-Yun Chen, Xiao-Pei Tian, Yu te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval, 2003.

[33] Xiaobai Chen, Aleksey Golovinskiy, , and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[34] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10:145–155, April 1992.

[35] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, SCG '05, pages 263–271, New York, NY, USA, 2005. ACM.

[36] Kree Cole-McLaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Loops in reeb graphs of 2-manifolds. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 344–350, New York, NY, USA, 2003. ACM.

[37] Nicu D. Cornea, Deborah Silver, Xiaosong Yuan, and Raman Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer*, 21, 2005.

[38] Fernando de Goes, Siome Goldenstein, and Luiz Velho. A hierarchical segmentation of articulated bodies. *Comput. Graph. Forum*, 27(5):1349–1356, 2008.

[39] Mathieu Desbrun, Anil N. Hirani, Melvin Leok, and Jerrold E. Marsden. Discrete exterior calculus. http://arxiv.org/abs/math?papernum=0508341, Aug 2005.

[40] Tamal K. Dey, K. Li, Chuanjiang Luo, Pawas Ranjan, Issam Safa, and Yusu Wang. Persistent heat signature for pose-oblivious matching of incomplete models. *Comput. Graph. Forum*, 29(5):1545–1554, 2010.

[41] Tamal K. Dey and Jian Sun. Defining and computing curve-skeletons with medial geodesic function. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 143–152, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[42] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des.*, 22(5):392–423, 2005.

[43] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066, 2006.

[44] H Edelsbrunner and J Harer. Persistent homology-a survey. *Contemporary Mathematics*, 453:1–26, 2008.

[45] Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *J. Math. Imaging Vis.*, 27(3):203–216, 2007.

[46] A Elad and Ron Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[47] Patrizio Frosini and Claudia Landi. "size functions and formal series". *Appl Algebra Engrg Comm Comput*, 12(4):327–349, 2001.

[48] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Trans. Graph.*, 22(1):83–105, 2003.

[49] Thomas Funkhouser and Philip Shilane. Partial matching of 3D shapes with priority-driven search. In *Symposium on Geometry Processing*, June 2006.

[50] Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25:130–150, January 2006.

[51] Ran Gal, Ariel Shamir, and Daniel Cohen-Or. Pose-oblivious shape signature. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):261–271, 2007.

[52] Ran Gal, Ariel Shamir, Tal Hassner, Mark Pauly, and Daniel Cohen-Or. Surface reconstruction using local shape priors. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 253–262, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[53] Timothy Gatzke, Cindy Grimm, Michael Garland, and Steve Zelinka. Curvature maps for local shape comparison. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 246–255, Washington, DC, USA, 2005. IEEE Computer Society.

[54] Katarzyna Gebal, Jakob Andreas Bærentzen, Henrik Aanæs, and Rasmus Larsen. Shape analysis using the auto diffusion function. *Comput. Graph. Forum*, 28(5):1405–1413, 2009.

[55] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Symposium on Geometry Processing*, pages 197–206, 2005.

[56] Joachim Giesen, Balint Miklos, Mark Pauly, and Camille Wormser. The scale axis transform. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 106–115, New York, NY, USA, 2009. ACM.

[57] D. Giorgi, P. Frosini, M. Spagnuolo, and B. Falcidieno. 3d relevance feedback via multilevel relevance judgements. *Vis. Comput.*, 26:1321–1338, October 2010.

[58] Afzal Godil, Helin Dutagaci, Ceyhun Burak Akgül, Apostolos Axenopoulos, Benjamin Bustos, Mohamed Chaouch, Petros Daras, Takahiko Furuya, Sebastian Kreft, Zhouhui Lian, Thibault Napoleon, Athanasios Mademlis, Ryutarou Ohbuchi, Paul L. Rosin, Bülent Sankur, Tobias Schreck, Xianfang Sun, Masaki Tezuka, Anne Verroust-Blondet, M. Walter, and Yücel Yemez. Shrec'09 track: Generic shape retrieval. In Michela Spagnuolo, Ioannis Pratikakis, Remco C. Veltkamp, and Theoharis Theoharis, editors, *3DOR*, pages 61–68. Eurographics Association, 2009.

[59] Aleksey Golovinskiy and Thomas Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 27(5), December 2008.

[60] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, New York, NY, USA, 2001. ACM.

[61] Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. Curve analogies. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 233–246, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[62] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM Press.

[63] Berthold K. P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(2):1671–1686, 1984.

[64] Qixing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape decomposition using modal analysis. *Computer Graphics Forum*, 28(2), 2009. to appear.

[65] Adrian Ion, Nicole M. Artner, Gabriel Peyrï¿½, Salvador B. López Mármol, Walter G. Kropatsch, and Laurent Cohen. 3d shape matching by geodesic eccentricity. In *Computer Vision and Pattern Recognition Workshops, 2008*, pages 1 – 8. IEEE Computer Society, June 2008.

[66] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:1254–1259, November 1998.

[67] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[68] Varun Jain and Hao Zhang. A spectral approach to shape-based retrieval of articulated 3d models. *Comput. Aided Des.*, 39:398–407, May 2007.

[69] Varun Jain, Hao Zhang, and Oliver van Kaick. Non-rigid spectral correspondence of triangle meshes. *International Journal on Shape Modeling*, 13(1):101–124, 2007.

[70] Miao Jin, Yalin Wang, Shing-Tung Yau, and Xianfeng Gu. Optimal global conformal surface parameterization. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 267–274, Washington, DC, USA, 2004. IEEE Computer Society.

[71] Tamal Dey Joachim, Joachim Giesen, and Samrat Goswami. Shape segmentation and matching with flow discretization. In *In Proc. Workshop on Algorithms and Data Structures*, pages 25–36, 2003.

[72] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[73] Aradrew E. Johnson. Surface landmark selection and matching in natural terrain. In *In Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 413–420, 2000.

[74] E Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(4):1, 2010.

[75] Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8–10):649–658, 2005.

[76] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, July 2003.

[77] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, June 2003.

[78] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Shape matching and anisotropy. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, August 2004.

[79] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Symmetry descriptors and 3d shape matching. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 115–123, New York, NY, USA, 2004. ACM.

[80] Hyun Soo Kim, Han Kyun Choi, and Kwan H. Lee. Feature detection of triangular meshes based on tensor voting theory. *Comput. Aided Des.*, 41:47–58, January 2009.

[81] Vladimir G. Kim, Yaron Lipman, and Tom Funkhouser. Blended intrinsic maps. *Transactions on Graphics (Proc. of SIGGRAPH 2011)*, 2011.

[82] Jan J. Koenderink and Andrea J. van Doorn. Surface shape and curvature scales. *Image Vision Comput.*, 10(8):557–565, 1992.

[83] Hamid Laga. Semantics-driven approach for automatic selection of best views of 3d shapes. In Mohamed Daoudi, Tobias Schreck, Michela Spagnuolo, Ioannis Pratikakis, Remco C. Veltkamp, and Theoharis Theoharis, editors, *3DOR*, pages 15–22. Eurographics Association, 2010.

[84] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Trans. Graph.*, 24(3):659–666, 2005.

[85] Richard B. Lehoucq, Danny C. Sorensen, and C. Yang. *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicityly Restorted Arnoldi Methods (Software, Environments, Tools)*. Soc for Industrial & Applied Math, 2009.

[86] Bruno Levy. Laplace-beltrami eigenfunctions: Towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications, invited talk*, 2006.

[87] Xiaolan Li, Afzal Godil, and Asim Wagan. Spatially enhanced bags of words for 3d shape retrieval. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Paolo Remagnino, Fatih Porikli, Jörg Peters, James Klosowski, Laura Arns, Yu Chun, Theresa-Marie Rhyne, and Laura

Monroe, editors, *Advances in Visual Computing*, volume 5358 of *Lecture Notes in Computer Science*, pages 349–358. Springer Berlin / Heidelberg, 2008.

[88] Zhouhui Lian, Paul L. Rosin, and Xianfang Sun. Rectilinearity of 3d meshes. *Int. J. Comput. Vision*, 89:130–151, September 2010.

[89] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):286–299, 2007.

[90] Rong Liu and Hao Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics 2007)*, 26(3):385–394, 2007.

[91] A. Mademlis, P. Daras, D. Tzovaras, and M. G. Strintzis. On 3D Partial Matching of Meaningful Parts. In *Image Processing*, volume 2, pages 517–520, 2007.

[92] Athanasios Mademlis, Petros Daras, Dimitrios Tzovaras, and Michael G. Strintzis. 3d object retrieval using the 3d shape impact descriptor. *Pattern Recognition*, pages 2447–2459, 2009.

[93] Siddharth Manay, Byung-Woo Hong, Anthony J. Yezzi, and Stefano Soatto. Integral invariant signatures. In Tomás Pajdla and Jiri Matas, editors, *ECCV (4)*, volume 3024 of *Lecture Notes in Computer Science*, pages 87–99. Springer, 2004.

[94] Simone Marini, Silvia Biasotti, and Bianca Falcidieno. Partial matching by structural descriptors. In *Content-Based Retrieval*, 2006.

[95] Diana Mateus, Radu P. Horaud, David Knossow, Fabio Cuzzolin, and Edmond Boyer. Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[96] G.J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

[97] S. Minakshisundaram. A uniqueness theorem for eigenfunction expansions. *Proceedings of the National Academy of Sciences*, 33(4):76–77, 1947.

[98] N. J. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):560–568, 2006.

[99] Niloy J. Mitra, Leonidas Guibas, Joachim Giesen, and Mark Pauly. Probabilistic fingerprints for shapes. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 121–130, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[100] Facundo Mémoli. Gromov-hausdorff distances in euclidean spaces. In *In Proc. Computer Vision and Pattern Recognition (CVPR)*, 2004.

[101] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. In *SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications*, pages 339–344, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[102] Michela Mortara, Giuseppe Patanè, Michela Spagnuolo, Bianca Falcidieno, and Jarek Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2003.

[103] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Spectral conformal parameterization. *Computer Graphics Forum*, 27(5):1487–1494, July 2008.

[104] Gelfand N. and L. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symp. Geom. Processing*, 2004.

[105] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, New York, NY, USA, 2006. ACM.

[106] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. *Computer Graphics Forum*, 30(5):1481–1491, 2011.

[107] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 613–622, New York, NY, USA, 2004. ACM.

[108] Marcin Novotni, Patrick Degener, and Reinhard Klein. Correspondence generation and matching of 3d shape subparts. Technical Report CG-2005-2, Universität Bonn, June 2005.

[109] Marcin Novotni and Reinhard Klein. Shape retrieval using 3d zernike descriptors. *Computer Aided Design*, 36(11):1047–1062, 2004.

[110] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3d models with shape distributions. *Shape Modeling and Applications, International Conference on*, 0:0154, 2001.

[111] Maks Ovsjanikov, Alexander M. Bronstein, Michael M. Bronstein, and Leonidas J. Guibas. Shapegoogle: a computer vision approach for invariant shape retrieval. In *Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment (ICCV workshop, NORDIA'09)*, october 2009.

[112] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J. Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Transactions on Graphics*, 30(4):33, 2011.

[113] Maks Ovsjanikov, Quentin Mérigot, Facundo Mémoli, and Leonidas J. Guibas. One point isometric matching with the heat kernel. *Comput. Graph. Forum*, 29(5):1555–1564, 2010.

[114] Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. Global intrinsic symmetries of shapes. *Computer Graphics Forum*, 27(5):1341–1348, July 2008.

[115] G. Patane, M. Spagnuolo, and B. Falcidieno. Reeb graph computation based on a minimal contouring. *2008 IEEE International Conference on Shape Modeling and Applications*, pages 73–82, 2008.

[116] Mark Pauly, Niloy J. Mitra, Joachim Giesen, Markus Gross, and Leonidas J. Guibas. Example-based 3d scan completion. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 23, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[117] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[118] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[119] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

[120] Helmut Pottmann, Johannes Wallner, Qixing Huang, and Yong-Liang Yang. Integral invariants for robust geometry processing. *Comput. Aided Geom. Design*, 26:37–60, 2009.

[121] Martin Reuter. *Laplace Spectra for Shape Recognition.* Books on Demand GmbH, 2006.

[122] Martin Reuter, Silvia Biasotti, Daniela Giorgi, Giuseppe Patanè, and Michela Spagnuolo. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390, 2009.

[123] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-spectra as fingerprints for shape matching. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 101–106, New York, NY, USA, 2005. ACM.

[124] Steven Rosenberg. *The Laplacian on a Riemannian Manifold*. Number 31 in London Mathematical Society Student Texts. Cambridge University Press, 1997.

[125] Raif M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 225–233, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[126] Raif M. Rustamov. A versatile framework for shape description. *Vis. Comput.*, 26:1245–1256, October 2010.

[127] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[128] Ariel Shamir, Lior Shapira, and Daniel Cohen-Or. Mesh analysis using geodesic mean-shift. *Vis. Comput.*, 22(2):99–108, 2006.

[129] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3d objects. *Int. J. Comput. Vision*, 89:309–326, September 2010.

[130] Andrei Sharf, Thomas Lewiner, Ariel Shamir, and Leif Kobbelt. On-the-fly curve-skeleton computation for 3d shapes. *Comput. Graph. Forum*, pages 323–328, 2007.

[131] Yonggang Shi, Rongjie Lai, Sheila Krishna, Nancy Sicotte, Ivo Dinov, and Arthur W Toga. Anisotropic laplace-beltrami eigenmaps: Bridging reeb graphs and skeletons. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*, 2008:1–7, 2008.

[132] Philip Shilane and Thomas Funkhouser. Selecting distinctive 3D shape descriptors for similarity retrieval. In *Shape Modeling International*, June 2006.

[133] Philip Shilane and Thomas Funkhouser. Distinctive regions of 3D surfaces. *ACM Transactions on Graphics*, 26(2):Article 7, June 2007.

[134] Takafumi Shimizu, Hiroaki Date, Satoshi Kanai, and Takeshi Kishinami. A new bilateral mesh smoothing method by recognizing features. *Computer Aided Design and Computer Graphics, International Conference on*, 0:281–286, 2005.

[135] Kaleem Siddiqi, Ali Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 222, Washington, DC, USA, 1998. IEEE Computer Society.

[136] Josef Sivic and Andrew Zisserman. Video google: Efficient visual search of videos. In Jean Ponce, Martial Hebert, Cordelia Schmid, and Andrew Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2006.

[137] P. Skraba, M. Ovsjanikov, F. Chazal, and L. Guibas. Persistence-based segmentation of deformable shapes. In *CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, page to appear, June 2010.

[138] Justin Solomon, Mirela Ben-Chen, Adrian Butscher, and Leonidas Guibas. Discovery of intrinsic primitives on triangle meshes. In *Proc. Eurographics 2011*, 2011.

[139] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM.

[140] D. Steiner and A. Fischer. Topology recognition of 3d closed freeform objects based on topological graphs. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, page 82, Washington, DC, USA, 2001. IEEE Computer Society.

[141] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 399–405, New York, NY, USA, 2004. ACM.

[142] Jian Sun, Xiaobai Chen, and Thomas Funkhouser. Fuzzy geodesics and consistent sparse correspondences for deformable shapes. *Computer Graphics Forum (Symposium on Geometry Processing)*, 29(5), July 2010.

[143] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1383–1392, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[144] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 130, Washington, DC, USA, 2003. IEEE Computer Society.

[145] Johan W. H. Tangelder and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl.*, 39(3):441–471, 2008.

[146] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation, 1995.

[147] Joshua B. Tenenbaum and William T. Freeman. Separating style and content with bilinear models. *Neural Comput.*, 12:1247–1283, June 2000.

[148] Julien Tierny. *Reeb graph based 3D shape modeling and applications.* PhD thesis, TELECOM Lille 1, Universite des Sciences et Technologies de Lille, 2008.

[149] Julien Tierny, Jean-Philippe Vandeborre, and Mohamed Daoudi. Topology driven 3D mesh hierarchical segmentation. In *IEEE International Conference on Shape Modeling and Applications (SMI 2007)*, pages 215–220, Lyon, France, 2007.

[150] Julien Tierny, Jp Vandeborre, and Mohamed Daoudi. Partial 3d shape retrieval by reeb pattern unfolding. *Computer Graphics Forum*, 28(1):41–55, 2009.

[151] Tony Tung and Francis Schmitt. The augmented multiresolution reeb graph approach for content-based retrieval of 3d shapes. *International Journal of Shape Modeling*, 11(1):91–120, 2005.

[152] D. V. Vranic and D. Saupe. 3D Shape Descriptor Based on 3D Fourier Transform. In *Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001) (editor K. Fazekas)*, pages 271–274, Budapest, Hungary, September 2001.

[153] Dejan V. Vranic. 3d model retrieval, 2001.

[154] Dejan V. Vranic. An improvement of rotation invariant 3d-shape based on functions on concentric spheres. In *ICIP (3)*, pages 757–760, 2003.

[155] Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun. Discrete laplace operators: no free lunch. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 33–37, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[156] T. Weinkauf and D.Günthernther. Separatrix persistence: Extraction of salient edges on surfaces using topological methods. *Computer Graphics Forum (Proc. SGP '09)*, 28(5):1519–1528, July 2009.

[157] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 505–512, 2002.

[158] Guoliang Xu. Discrete laplace–beltrami operators and their convergence. *Comput. Aided Geom. Des.*, 21:767–784, October 2004.

[159] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhiquan Cheng. Style-content separation by anisotropic part scales. *ACM Transactions on Graphics, (Proc. of SIGGRAPH Asia 2010)*, 29(5):to appear, 2010.

[160] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Yueshan Xiong. Dynamic harmonic fields for surface processing. *Computers and Graphics (Special Issue of Shape Modeling International)*, 33:391–398, 2009.

[161] Kai Xu, Hao Zhang, Andrea Tagliasacchi, Ligang Liu, Guo Li, Min Meng, and Yueshan Xiong. Partial intrinsic reflectional symmetry of 3d shapes. *ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2009)*, 28(5):Article 138, 2009.

[162] Christopher Zach, Manfred Klopschitz, and Manfred Pollefeys. Disambiguating visual relations using loop constraints. In *CVPR'10*, pages 1426–1433, 2010.

[163] Cha Zhang and Tsuhan Chen. Efficient feature extraction for 2d/3d objects in mesh representation. In *ICIP (3)*, pages 935–938, 2001.

[164] Hao Zhang, Alla Sheffer, Daniel Cohen-Or, Qingnan Zhou, Oliver van Kaick, and Andrea Tagliasacchi. Deformation-drive shape correspondence. *Computer Graphics Forum (Special Issue of Symposium on Geometry Processing 2008)*, 27(5):1431–1439, 2008.

[165] Hao Zhang, Oliver van Kaick, and Ramsay Dyer. Spectral methods for mesh processing and analysis. In *Proc. of Eurographics State-of-the-art Report*, pages 1–22, 2007.